

**Diese gekürzte Fassung beinhaltet die Planung,  
Ergebnisse der Auswertung und Arbeitsblätter.**

**Behandlung von Primzahltests unter der Benutzung eines Computeralgebra-Systems  
als didaktisches Hilfsmittel und kognitives Werkzeug in einer 11. Klasse  
(Förderklasse Mathematik) des Gymnasiums**

Hausarbeit im Rahmen der Zweiten Staatsprüfung für das Lehramt an Gymnasien

vorgelegt von

**Jens Bernheiden**

Greifswald, den 01.08.2001

**"Instead of collecting (...) bottle caps I collect  
prime numbers (...) They're just as rare."<sup>1</sup>**

Ich danke allen, die mir während der Entstehung dieser Arbeit  
ihre Unterstützung zukommen ließen.

Mein besonders herzlicher Dank gilt Frau Albrecht, Uta Appelt,  
Johanna und Jörg Bernheiden, Sven Bernheiden, Matthias Döppe,  
Frau Hartwig, Frau Heinlein, Frau Lotz, Herrn Prof. Dr. Schlosser  
und den Schülerinnen und Schülern der Klasse 11M.

---

<sup>1</sup> Zitat von Roland Clarkson, der als neunzehnjähriger Student die 37. Mersenneschen Primzahl entdeckte.  
(Johnson, Tracy: Student Finds Largest Prime Number Ever. In: Los Angeles Times, v. 06.02.1998.  
In: <http://members.aol.com/m3021377/private/latimes.html>).

# Inhaltsverzeichnis

<b>1. EINLEITUNG</b>	<b>4</b>
1. 1. WARUM PRIMZAHLTESTS?	4
1. 2. ZIELE DES UNTERRICHTSVERSUCHES	5
<b>2. PLANUNG UND ANALYSE DES UNTERRICHTSVERSUCHES</b>	<b>6</b>
2. 1. RAHMENBEDINGUNGEN	6
2. 2. SACHANALYSE	7
2. 2. 1. PRIMZAHLEN	7
2. 2. 2. PRIMZAHLTESTS	7
2. 3. DIDAKTISCHE ANALYSE	9
2. 3. 1. BILDUNGSGEHALT DES THEMAS	9
2. 3. 2. DIDAKTISCHE REDUKTION	10
2. 4. ANGESTREBTE UNTERRICHTSERGEBNISSE	11
2. 5. METHODISCHE ANALYSE	12
2. 5. 1. MOTIVATION	12
2. 5. 2. VOLLSTÄNDIGKEIT UND AUSFÜHRLICHKEIT	12
2. 5. 3. LOGISCHE ABFOLGE DER INHALTE – EINFÜHRUNG DES KONGRUENZBEGRIFFS	12
2. 5. 4. LEHRERVORTRAG UND UNTERRICHTSGESPRÄCH	13
2. 5. 5. BEWEISE	13
2. 5. 6. DAS COMPUTERALGEBRA-SYSTEM MUPAD	13
2. 5. 7. KONZEPTION DER VOM CAS – EINSATZ GEPRÄGTEN STUNDEN	14
2. 5. 8. PRIMZAHLTESTS UND IHRE ERARBEITUNG MIT EINEM CAS	14
2. 5. 9. KONTROLLE DER LERNERGEBNISSE	15
2. 6. VERLAUFSPLANUNG	16
<b>3. AUSWERTUNG DES UNTERRICHTSVERSUCHES</b>	<b>FEHLER! TEXTMARKE NICHT DEFINIERT.</b>
3. 1. KRITISCHE REFLEXION DER DURCHFÜHRUNG	FEHLER! TEXTMARKE NICHT DEFINIERT.
3. 1. 1. ERSTE UND ZWEITE STUNDE	FEHLER! TEXTMARKE NICHT DEFINIERT.
3. 1. 2. DRITTE STUNDE	FEHLER! TEXTMARKE NICHT DEFINIERT.
3. 1. 3. VIERTE UND FÜNFTE STUNDE	FEHLER! TEXTMARKE NICHT DEFINIERT.
3. 1. 4. SECHSTE STUNDE	FEHLER! TEXTMARKE NICHT DEFINIERT.
3. 1. 5. SIEBENTE UND ACHTE STUNDE	FEHLER! TEXTMARKE NICHT DEFINIERT.
3. 1. 6. NEUNTE STUNDE	FEHLER! TEXTMARKE NICHT DEFINIERT.
3. 1. 7. ZEHNTE UND ELFTE STUNDE	FEHLER! TEXTMARKE NICHT DEFINIERT.
3. 1. 8. ZWÖLFTE STUNDE	FEHLER! TEXTMARKE NICHT DEFINIERT.
3. 2. ERREICHTE UNTERRICHTSERGEBNISSE	FEHLER! TEXTMARKE NICHT DEFINIERT.
3. 3. BEWERTUNG DER PLANUNG UND ANSÄTZE ZUR VERBESSERUNG	18
3. 4. ZAHLENTHEORIE IN DER GYMNASIALEN OBERSTUFE	19
3. 5. DER CAS – EINSATZ IM UNTERRICHT	20
3. 6. AUSBAU DES THEMAS	21
<b>4. ZUSAMMENFASSUNG</b>	<b>22</b>
<b>5. LITERATURVERZEICHNIS</b>	<b>23</b>
5. 1. BÜCHER, ZEITSCHRIFTEN UND BILDUNGSPLÄNE	23
5. 2. QUELLEN AUS DEM INTERNET	24
5. 3. TITELBILD	25
<b>6. SYMBOLE UND BEZEICHNUNGEN</b>	<b>26</b>
<b>7. ANHANG</b>	<b>27</b>
7. 1. INHALT DER BEILIEGENDEN CD-ROM	27
7. 2. ARBEITSBLÄTTER, FOLIEN UND ERGÄNZUNGEN	28

# 1. Einleitung

## 1. 1. Warum Primzahltests?

Im Jahr 1603 behauptete der Franziskanermönch Marin Mersenne<sup>2</sup>, dass für alle Primzahlen  $p$  bis 257 nur die Zahlen 2, 3, 5, 7, 13, 17, 19, 31, 67, 127 und 257 in der Formel  $2^p - 1$  Primzahlen liefern. Ungeachtet dessen, dass  $2^{67} - 1$  und  $2^{257} - 1$  zusammengesetzt sind und Mersenne die Zahlen 61, 89 und 107 übersehen hatte, was veranlasste den Mönch und andere Gelehrte, wie z.B. Fermat<sup>3</sup>, Descartes<sup>4</sup>, Euler<sup>5</sup> und Gauß<sup>6</sup>, sich mit Primzahlen zu beschäftigen? Sicher war es in erster Linie die Mystik dieser Zahlen: „Das vielleicht größte Rätsel der Primzahlen besteht darin, dass sie sich – ungeachtet ihrer einfachen Definition – ziemlich irregulär verhalten.“<sup>7</sup>

Das anscheinend regelwidrige Auftreten der unzerlegbaren Zahlen ist heute, im Zeitalter der modernen Rechentechnik, dafür verantwortlich, dass immer mehr Menschen nach großen Primzahlen «jagen». Die rasant fortschreitenden Entwicklungen in der Sicherheit der Übermittlung von Informationen belegen, dass Hardy<sup>8</sup> sich irrte, als er äußerte: „Die ‚echte‘ Mathematik der ‚echten‘ Mathematiker, die Mathematik von Fermat, Gauß, Abel und Riemann ist fast völlig nutzlos.“<sup>9</sup>

Im Unterschied zum 17. Jahrhundert, als Mersenne die 78-stellige Zahl  $2^{257} - 1$  als Primzahl einstufte, geht es gegenwärtig um Zahlen mit mehr als einer Million Stellen. Das Testen solcher riesigen Zahlen auf die Primzahleigenschaft dauert selbst mit Hochleistungsrechnern sehr lange. Effiziente Primzahltests sind aktueller denn je, und daher sollte man Jugendlichen einen Einblick in dieses Themengebiet nicht verwehren.

---

<sup>2</sup> Marin Mersenne (1588 – 1648).

<sup>3</sup> Pierre de Fermat (1601 – 1665).

<sup>4</sup> René Descartes (1596 – 1650).

<sup>5</sup> Leonhard Euler (1707 – 1783).

<sup>6</sup> Carl Friedrich Gauß (1777 – 1855).

<sup>7</sup> Conway 1997, 143.

<sup>8</sup> Godefrey Harold Hardy (1877 – 1947).

<sup>9</sup> <http://www.mathe.tu-freiberg.de/~hebisch/cafe/zitate.html>.

## 1. 2. Ziele des Unterrichtsversuches

Mit dem gewählten Thema wird ein mathematischer Gegenstand in den Vordergrund des Unterrichts gerückt, den die Schulmathematik recht vernachlässigt. Der Unterrichtsversuch soll daher untersuchen:

- Sind Primzahltests als Teilgebiet der Zahlentheorie für den Unterricht der gymnasialen Oberstufe, insbesondere in einer 11. Klasse geeignet?

Der Computer soll in den Mathematikstunden als multimediales Instrument Aufgaben tradierter Medien übernehmen. Außerdem wird ein Computeralgebra-System als didaktisches Hilfsmittel und kognitives Werkzeug eingesetzt. „An geeigneten Stellen sind den Schülern algorithmische Vorgehensweisen bewusst zu machen (...) Die Nutzung aktueller Software für einzelne Bereiche der Mathematik muss zur selbstverständlichen Arbeitsweise im Mathematikunterricht gehören.“<sup>10</sup>

Computer, insbesondere Computeralgebra-Systeme, finden erst seit kurzem im Unterricht Verwendung. Es gibt daher noch zu wenige Praxiserfahrungen. Deshalb soll der Unterrichtsversuch helfen, folgende Fragen zu klären:

- Ist das Thema «Primzahltests» für den sinnvollen Einsatz eines Computeralgebra-Systems geeignet, und inwieweit hilft das CAS<sup>11</sup> den Schülern<sup>12</sup> bei der Erarbeitung neuer Sachverhalte?
- Welche Probleme können bei der Arbeit mit einem Computeralgebra-System auftreten?
- Können die Lernenden durch den Einsatz des Computers als interaktives Medium und durch die Verwendung eines Computeralgebra-Systems zur Auseinandersetzung mit mathematischen Sachverhalten angespornt werden?

---

<sup>10</sup> Rahmenplan: Sekundarstufe II. 1999.

<sup>11</sup> Computeralgebra-System.

<sup>12</sup> Mit dem Ausdruck «Schüler» sind im Folgenden Schülerinnen und Schüler gemeint, um den Lesefluss nicht unnötig zu erschweren.

## 2. Planung und Analyse des Unterrichtsversuches

### 2. 1. Rahmenbedingungen

...

Die Unterrichtseinheit «Primzahltests» wird gegen Ende des Schuljahres unterrichtet. Der Mathematikunterricht findet jeweils an zwei Wochentagen statt. Die Doppelstunde wird auf Grund der räumlichen Gegebenheiten in einer Nachbarschule erteilt. Der Einsatz des Computers ist nur in der Einzelstunde möglich, wobei immer zwei Lernende an einem Rechner arbeiten.

Alle Lernenden haben zu Hause Zugang zu einem PC<sup>13</sup>. Nach Absprache mit der Klasse waren die Schüler einstimmig dafür, das Computeralgebra-System MuPAD<sup>14</sup> auf ihren Rechnern zu installieren. Keinem der Lernenden ist das CAS bekannt gewesen, jedoch wurde in dieser Klasse schon andere Mathematiksoftware (z.B. DERIVE) eingesetzt. Bereitwillig bearbeiteten die Schüler zusätzlich zum Schulstoff Arbeitsblätter<sup>15</sup>, um die Anwendung des Computeralgebra-Systems zu erlernen.<sup>16</sup> Auftretende Probleme wurden in Pausen besprochen und weitestgehend gelöst. In der letzten Stunde vor dem Unterrichtsversuch hatten die Schüler die Möglichkeit, Fragen zur Arbeit mit MuPAD zu stellen. Außerdem konnten sie selbstgewählte Aufgaben mit dem CAS lösen. Das letzte MuPAD-Arbeitsblatt<sup>17</sup> beinhaltete Elemente der Zahlentheorie, so dass die Schüler sich mit einigen, im Unterrichtsversuch benötigten, fachlichen Grundlagen auseinandersetzen mussten.

---

<sup>13</sup> Personalcomputer.

<sup>14</sup> Multi Processing Algebra Data Tool.

<sup>15</sup> Vgl. MuPAD-Arbeitsblätter 1 bis 8. ANHANG, 36ff.

<sup>16</sup> Diese Tatsache stützt das schon beschriebene Interesse der Lernenden für Mathematik.

<sup>17</sup> Vgl. MuPAD-Arbeitsblatt 8. ANHANG, 46.

## 2. 2. Sachanalyse

### 2. 2. 1. Primzahlen

„Untersuchungen verschiedener Eigenschaften natürlicher Zahlen gehörten historisch zu den ältesten Beschäftigungen mit mathematischen Problemen überhaupt.“<sup>18</sup> Die *Teilbarkeitsrelation*<sup>19</sup> ist dabei die Grundlage der zahlentheoretischen Forschung: „Das unvergängliche Problem der Zahlentheorie ist das der Teilbarkeit: Ist eine Zahl durch eine andere teilbar oder nicht?“<sup>20</sup> Man erkennt sofort, dass sich einige Zahlen aus der Menge der natürlichen Zahlen herausheben: Es gibt Zahlen, die nur zwei Teiler besitzen. Bereits um 300 v. Chr. formulierte und bewies Euklid<sup>21</sup>: „Die Primzahlen sind mehr als jede vorgegebene Menge von Primzahlen.“<sup>22</sup> Mittlerweile gibt es mehr als 20 Beweise des Satzes von Euklid über die *Unendlichkeit der Primzahlmenge*.<sup>23</sup> Viele Mathematiker versuchten, eine Primzahlformel, die alle Primzahlen und nur diese beschreibt, oder doch wenigstens eine, die sehr viele Primzahlen und dabei wenige zusammengesetzte Zahlen liefert, zu finden. Genannt seien hier die *Fermatschen Zahlen*  $F_n = 2^{2^n} + 1$  und die *Mersenneschen Zahlen*  $M_n = 2^n - 1$ , wobei die letzteren heutzutage beim Finden großer Primzahlen eine zentrale Stellung einnehmen.

### 2. 2. 2. Primzahltests

Würde man eine einfache Gleichung kennen, die genau alle Primzahlen erzeugt, könnte man auf komplizierte Primzahltests verzichten. Schon lange beschäftigen sich Mathematiker mit der Frage, wie man möglichst schnell und sicher feststellen kann, ob eine vorliegende natürliche Zahl die Primzahleigenschaft erfüllt.

Sichere Tests dauern bei großen Zahlen sehr lange und sind deshalb unpraktikabel. Effizienter sind stochastische Testverfahren<sup>24</sup>, die z.B. auf dem *Kleinen Satz von Fermat* oder dem *Satz von Wilson*<sup>25</sup> basieren. Der hier zu entwickelnde Primzahltest bezieht sich auf den *Kleinen Satz von Fermat*, der als Spezialfall im *Eulerschen Satz* enthalten ist. Zur Formulierung des *Eulerschen Satzes* benötigt man die *Eulersche  $\varphi$ -Funktion*:

**Definition 1:** Sei  $m \in \mathbb{N}$ . Die Funktion  $\varphi: \mathbb{N} \rightarrow \mathbb{N}$  mit  $\varphi(m) =$  Anzahl der zu  $m$  teilerfremden Zahlen aus  $\{1; 2; \dots; m\}$  heißt *Eulersche  $\varphi$ -Funktion*.

$\varphi(m)$  beschreibt demnach auch die Anzahl der *primen Restklassen modulo  $m$* <sup>26</sup> und die Anzahl der Zahlen  $a \in \mathbb{N}$  mit  $1 \leq a \leq m$  und  $\text{ggT}(a, m) = 1$ .

**Satz 1:** (*Satz von Euler*)<sup>27</sup>

Seien  $a, m \in \mathbb{N}$ . Sind  $a$  und  $m$  teilerfremd, so gilt:  $a^{\varphi(m)} \equiv 1 \pmod{m}$ .

Die *primen Restklassen modulo  $m$*  bilden eine Gruppe. Die Gruppentheorie liefert: Gruppenelement<sup>Gruppenordnung</sup> = Einselement.<sup>28</sup> Also gilt  $\bar{a}^{\varphi(m)} = \bar{1}$  und daraus folgt sofort der *Satz von Euler*.<sup>29</sup> Für Primzahlen  $p$  ergibt sich  $\varphi(p) = p - 1$ .<sup>30</sup> Der *Eulersche Satz* geht in den *Kleinen Satz von Fermat* über:

**Satz 2:** (*Kleiner Satz von Fermat*)

Sei  $a \in \mathbb{N}$  und  $p \in \mathbb{N}$  eine Primzahl. Ist  $a$  kein Vielfaches von  $p$ , so gilt:  $a^{p-1} \equiv 1 \pmod{p}$ .

Aus dem *Kleinen Satz von Fermat* kann man sofort ableiten:

**Satz 3:** (*Folgerung aus dem Kleinen Satz von Fermat*)

Seien  $a, n \in \mathbb{N}$ . Gibt es ein  $a$  mit  $\text{ggT}(a, n) = 1$  und  $a^{n-1} \not\equiv 1 \pmod{n}$ , dann ist  $n$  zusammengesetzt.

<sup>18</sup> Bundschuh 1998, V.

<sup>19</sup> Um die für diese Arbeit wichtigsten mathematischen Fachausdrücke hervorzuheben, werden sie *kursiv* gekennzeichnet.

<sup>20</sup> Remmert 1995, 5.

<sup>21</sup> Euklid von Alexandria (um 300 v. Chr.).

<sup>22</sup> Remmert 1995, 25.

<sup>23</sup> Vgl. Ribenboim 1996, 3ff.; <http://www.theory-of-numbers.de/>.

<sup>24</sup> In der englischsprachigen Literatur findet man "probabilistic or Monte Carlo tests" als Fachtermini. (Vgl. Ribenboim 1996, 139).

<sup>25</sup> John Wilson (1741 – 1793).

<sup>26</sup> Seien  $a, b \in \mathbb{Z}$  und  $m \in \mathbb{N}$ .  $a$  heißt *kongruent  $b$  modulo  $m$* , genau dann, wenn gilt:  $m \mid (a - b)$ . Diese Relation ist *reflexiv*, *transitiv* und *symmetrisch* (Vgl. Bundschuh 1998, 79) und damit eine *Äquivalenzrelation auf  $\mathbb{Z}$* , die eine Klasseneinteilung der ganzen Zahlen bewirkt. Die Menge  $\bar{a} = \{b \mid b \in \mathbb{Z} \text{ und } b \equiv a \pmod{m}\}$  heißt *Restklasse modulo  $m$* .  $\bar{a}$  heißt *prime Restklasse modulo  $m$* , wenn  $a$  und  $m$  teilerfremd sind.

<sup>27</sup> Speziell auf den *Eulerschen Satz* bezogene Beweise findet man u.a. in Padberg 1999a, 69 und in Ribenboim 1996, 22. Letzterer wird mittels der *vollständigen Induktion* geführt.

<sup>28</sup> Vgl. Remmert 1995, 219.

<sup>29</sup> Die *prime Restklassengruppe modulo  $m$*  besitzt genau  $\varphi(m)$  Elemente (Vgl. Remmert 1995, 216).

<sup>30</sup> Alle natürlichen Zahlen kleiner  $p$  sind zu  $p$  teilerfremd.

Interessant für Primzahltests ist nun aber gerade die Umkehrung des *Kleinen Satzes von Fermat*. Diese gilt jedoch nicht: Es gibt zusammengesetzte Zahlen  $n$ , so dass für alle  $a \in \mathbb{N}$  mit  $\text{ggT}(a, n) = 1$  die Kongruenz  $a^{n-1} \equiv 1 \pmod{n}$  erfüllt ist; sie heißen *Carmichaelzahlen*.<sup>31</sup>

**Definition 2:** Sei  $a \in \mathbb{N}$  und  $n \in \mathbb{N}$  zusammengesetzt.

Gilt für alle  $a$  mit  $\text{ggT}(a, n) = 1$ :  $a^{n-1} \equiv 1 \pmod{n}$ , so heißt  $n$  *Carmichaelzahl*.

Weil es viel weniger *Carmichaelzahlen* als Primzahlen gibt, wäre ein ziemlich sicherer Primzahltest mit der Umkehrung des *Kleinen Satzes von Fermat* möglich.<sup>32</sup> Bei einer Primzahl bzw. einer Carmichaelzahl ist dann aber die  $(n-3)$ -malige Berechnung des größten gemeinsamen Teilers<sup>33</sup> und in den Fällen, in denen  $a$  und  $n$  teilerfremd sind, außerdem die Überprüfung der Kongruenz nötig.<sup>34</sup> Es liegt gegenüber sicheren Tests keine Zeitersparnis vor. Betrachtet man nicht alle Zahlen  $a \in \mathbb{N}$ , sondern beispielsweise nur ein  $a$  ( $1 < a < n-1$ ), dann liefern sogenannte *Pseudoprime zur Basis  $a$* , ein falsches Testergebnis.<sup>35</sup>

**Definition 3:** Sei  $a \in \mathbb{N}$  und  $n$  eine ungerade zusammengesetzte Zahl.

Gibt es ein  $a$  mit  $a^{n-1} \equiv 1 \pmod{n}$  so heißt  $n$  *Pseudoprime zur Basis  $a$* .

Glücklicherweise existieren auch nur wenige *Pseudoprime*.<sup>36</sup> Wenn man mehrere Basen in den Test einbezieht, wird das Ergebnis sehr sicher<sup>37</sup>, weil eine falsche Testentscheidung nur vorliegt, wenn  $n$  eine *Pseudoprime* zu allen getesteten Basen ist. Solch ein Pseudoprime-Test<sup>38</sup> ist sehr schnell, weil die Anzahl der getesteten Basen bestimmt, wie oft im schlechtesten Fall der größte gemeinsame Teiler berechnet und die Kongruenz überprüft werden muss. Bei der Testimplementierung wird man natürlich sofort die Testentscheidung «zusammengesetzt» ausgeben, wenn man ein  $a$  mit  $\text{ggT}(a, n) \neq 1$  gefunden hat, so dass man mit den Basen 2, 3 und 5 folgende Prüfabfolge erhält:

1. Ist  $n < 2$ , dann ist  $n$  keine Primzahl.
2. Ist  $n \in \{2; 3; 5\}$ , dann ist  $n$  eine Primzahl.
3. Gilt für  $a = 2, 3$  und  $5$ :  $\text{ggT}(a, n) = 1$  und  $a^{n-1} \equiv 1 \pmod{n}$ , dann ist  $n$  eine Primzahl.

Für eine *Pseudoprime*, die gleichzeitig *Pseudoprime* zu den Basen 2, 3 und 5 ist, wird es zu einer Fehlentscheidung kommen.

Man könnte einen Pseudoprime-Test sicher machen, indem man die zu untersuchende Zahl nach der Testentscheidung «Primzahl» noch mit einer Liste von *Pseudoprime* zu den geprüften Basen vergleicht.<sup>39</sup> Die Berechnung von *Pseudoprime* zu mehreren Basen benötigt bei großen Zahlen jedoch zu viele Operationen, so dass dies nicht praktikabel ist.

Primzahltests mit wahren Testentscheidungen, auch solche, die auf Kongruenzen basieren<sup>40</sup>, benötigen alle sehr viel Zeit, so dass in Computeralgebra-Systemen in der Regel stochastische Primzahltests genutzt werden, wobei ein ähnlicher Test, wie der beschriebene Pseudoprime-Test zu einigen Basen, ein Teil des Prüfverfahrens ist.

Beschreibungen der in aktuellen Computeralgebra-Systemen verwendeten Primzahltests lassen erkennen, dass ein Testverfahren meist wie folgt abläuft:<sup>41</sup>

1. Test auf Existenz eines kleinen Primfaktors<sup>42</sup>
2. Starker Pseudoprime-Test<sup>43</sup>
3. Primzahltest mittels Lucas-Folgen<sup>44</sup>

<sup>31</sup> Die Voraussetzung « $a$  kein Vielfaches von  $n$ » wurde zu « $\text{ggT}(a, n) = 1$ » verschärft.

<sup>32</sup> Anzahl der *Carmichaelzahlen*: Vgl. Riesel 1994, 94f.; Folie «Carmichaelzahlen». ANHANG, 63.

<sup>33</sup> Es reicht, alle Zahlen  $a \in \mathbb{N}$  mit  $1 < a < n-1$  zu überprüfen (Vgl. Umkehrung des *Kleinen Satzes von Fermat*. ANHANG, 62).

<sup>34</sup> Zeitkomplexität der Algorithmen zur Bestimmung des größten gemeinsamen Teilers von  $a$  und  $n$  und zur Berechnung des Restes von  $a^{n-1}$  bei Division durch  $n$ : Vgl. Ribenboim 1996, 137.

<sup>35</sup> Jede *Carmichaelzahl* ist eine *Pseudoprime*, jedoch ist eine *Carmichaelzahl* nicht *Pseudoprime* zu allen möglichen Basen. (Beispielsweise ist die kleinste *Carmichaelzahl* keine *Pseudoprime* zur Basis 3:  $3^{560} \equiv 375 \pmod{561}$ ). Diese Kongruenz wird innerhalb der Carmichaelzahldefinition nicht überprüft, weil  $\text{ggT}(3, 561) = 3$  ist.) Ein solcher Test entscheidet demnach für *Carmichaelzahlen*, die nicht *Pseudoprime zur Basis  $a$*  sind, richtig.

<sup>36</sup> Anzahl der *Pseudoprime*: Vgl. Riesel 1994, 94f.; Folie «Pseudoprime». ANHANG, 64.

<sup>37</sup> Vgl. Folie «Pseudoprime». ANHANG, 64.

<sup>38</sup> Mit «Pseudoprime-Test» wird der vorgestellte stochastische Primzahltest bezeichnet, nicht etwa ein Test, der entscheidet, ob eine vorliegende Zahl eine *Pseudoprime* ist (Vgl. Ribenboim 1996, 140).

<sup>39</sup> Ein Test von D. H. Lehmer arbeitete nach diesem Prinzip (Vgl. Riesel 1994, 86).

<sup>40</sup> Vgl. Ribenboim 1996, 48ff.

<sup>41</sup> Vgl. Primzahltests in aktuellen Computeralgebra-Systemen. ANHANG, 35.

<sup>42</sup> 76% aller ungeraden natürlichen Zahlen besitzen einen Primfaktor kleiner 100 (Vgl. Riesel 1994, 85).

<sup>43</sup> Bei einem starken Pseudoprime-Test (z.B. *Rabin-Miller-Primzahltest*) führen die noch selteneren *starken Pseudoprime* zu einer falschen Testentscheidung (Vgl. Bartholomé 1995, 164ff.; Riesel 1994, 91ff.).

<sup>44</sup> Primzahltests mittels Lucas-Folgen: Vgl. Dobermann 1999.

## 2. 3. Didaktische Analyse

### 2. 3. 1. Bildungsgehalt des Themas

In den unteren Jahrgangsstufen des Gymnasiums sind u.a. Grundbegriffe und Beziehungen der Teilbarkeit als verbindliche Inhalte des Unterrichts aufgeführt.<sup>45</sup> Im Zusammenhang mit der Bruchrechnung müssen die Schüler die Berechnung des größten gemeinsamen Teilers in allen Klassenstufen anwenden. Ansonsten findet die elementare Zahlentheorie, insbesondere die Primzahlthematik, kaum Berücksichtigung im Mathematikunterricht.

Wie ist dies zu erklären, wenn doch schon Gauß die Zahlentheorie als „Königin der Mathematik“<sup>46</sup> bezeichnete? In Fachdidaktikzeitschriften findet man immer wieder Artikel, in denen sich Autoren für die Behandlung zahlentheoretischer Themen im Unterricht aussprechen, z.B.:

„Im Mathematikunterricht finden Primzahlen wenig Berücksichtigung – und doch gibt es zahlreiche Möglichkeiten für eine Anbindung an den Lehrstoff und für selbständige Unterrichtseinheiten.“<sup>47</sup>

„Das Thema ‚Teilbarkeit‘ bietet interessante Problemstellungen, die zur Beschäftigung mit Mathematik anregen“<sup>48</sup>.

Die stoffliche Fülle der in der Jahrgangsstufe 11 zwingend zu behandelnden Themengebiete reicht in der zu unterrichtenden Klasse nicht aus.<sup>49</sup> Als Lehrer ist man gefordert, interessante Thematiken zu wählen, um diesen Schülern die breite Anwendbarkeit der Mathematik zu verdeutlichen.<sup>50</sup> Weshalb sollte man diesen leistungsstarken Lernenden nicht ein gegenwarts- und zukunftsbezogenes Thema<sup>51</sup> bieten, das heutzutage im Mathematikunterricht ungerechterweise verkümmert?

Ein großer Vorteil der elementaren Zahlentheorie ist, dass viele Fragestellungen sehr allgemeinverständlich formuliert werden können.<sup>52</sup> Die Aufgabenstellung kann in diesen Fällen somit durch jeden Schüler sofort erfasst werden. Die Lösung wird dagegen von den Lernenden eine aktive Auseinandersetzung mit den mathematischen Inhalten, Kritikfähigkeit, Kreativität, Selbständigkeit und Gewissenhaftigkeit erfordern.<sup>53</sup>

In der Zahlentheorie findet man viele Teilgebiete, die sich zu einer selbständigen Unterrichtseinheit ausbauen lassen. Beispielsweise könnte man den Schwerpunkt auf die Kryptographie legen, die Faktorisierung großer Zahlen untersuchen, das Erzeugen großer Primzahlen in den Vordergrund rücken oder aber Primzahltests thematisieren. Stellt man die aufgeführten Themen gegenüber, lassen sich jeweils Vorteile und Nachteile finden, die hier nicht erörtert werden sollen, jedoch muss der Unterricht in einer ersten Phase die gleichen wesentlichen fachlichen Grundlagen vermitteln.

Die Unterrichtseinheit «Primzahltests» ist geeignet, problemorientiert Erkenntnisprozesse zu initiieren, bietet Ansätze für historische und philosophische Betrachtungen und offenbart Möglichkeiten und Grenzen der Mathematik.<sup>54</sup> Die gewählte Thematik kann durch die Einbeziehung des Internets und den gezielten Einsatz eines CAS dazu beitragen, dass die Schüler „mit modernen Arbeitsmitteln und dazugehörigen Arbeitstechniken vertraut gemacht werden.“<sup>55</sup> Außerdem können durch die Verwendung eines Computeralgebra-Systems Beispiele herangezogen werden, die der Technikgläubigkeit<sup>56</sup> entgegenwirken.

<sup>45</sup> Vgl. Rahmenplan: Sekundarstufe I. 1997.

<sup>46</sup> Vgl. Remmert 1995, 5.

<sup>47</sup> Glatfeld 1993a, 4.

<sup>48</sup> Winning 1993, 18.

<sup>49</sup> So erfolgte beispielsweise innerhalb der Stochastik zusätzlich zu den vorgeschriebenen Inhalten die Erarbeitung von *zweiseitigen Signifikanztests* und ein Exkurs über *Schätzer*.

<sup>50</sup> Vgl. Forderungen des Rahmenplans (Rahmenplan: Sekundarstufe II. 1999).

<sup>51</sup> Große Primzahlen werden z.B. zur Erzeugung von Schlüsseln in der Kryptographie genutzt. Sichere Verschlüsselungsverfahren sind heute unverzichtbar und werden in der Zukunft noch wichtiger sein.

<sup>52</sup> Zum Beispiel ist die Problemstellung «Wie viele Primzahlen gibt es?» schnell formuliert und sofort durchschaubar.

<sup>53</sup> Die Aufzählung ist nicht erschöpfend und lässt sich fortsetzen.

<sup>54</sup> Vgl. Forderungen des Rahmenplans (Rahmenplan: Sekundarstufe II. 1999).

<sup>55</sup> Vgl. Rahmenplan: Sekundarstufe II. 1999.

<sup>56</sup> Immer schnellere Prozessoren werden entwickelt und in der Werbung angepriesen. Nicht wenige Jugendliche sind der Meinung, dass nur die Zahl vor dem «MHz» dafür verantwortlich ist, wie schnell ein Computer die Antwort auf eine gestellte Aufgabe findet; was den Rechner dazu befähigt, wird meist nicht erahnt. Bei der Behandlung von Primzahltests bieten sich Gelegenheiten, die Notwendigkeit der Entwicklung effizienter Algorithmen (und damit die Notwendigkeit mathematischer Grundlagenforschung) zu verdeutlichen und die Mathematik als herausragendes wissenschaftliches Instrument hervorzuheben.

### 2. 3. 2. Didaktische Reduktion

Die fachwissenschaftlichen Grundlagen, die den hier zu behandelnden Pseudoprimzahltests zu Grunde liegen, sind für die Schüler zum großen Teil neu. Sie werden für die meisten eine hohe intellektuelle Herausforderung und für manche auch eine Überforderung darstellen.

Eine zu enge Führung der Thematik ohne Berücksichtigung von interessanten Nebeninformationen erscheint für den Unterricht eher ungeeignet, weil die in Abschnitt 2.4. dargestellten Ziele so nicht erreicht werden können. Je mehr Sachverhalte man jedoch in den Unterricht aufnimmt, desto öfter muss man auf ein tieferes Hinterfragen dieser verzichten<sup>57</sup>. Eine kritische Auswahl der Inhalte und ein Kompromiss bei der Ausführlichkeit ihrer Behandlung erscheinen notwendig. Daher sollen die wesentlichen Kernpunkte sehr ausführlich diskutiert, einzelne Nebeninformationen aber vereinfacht bzw. verkürzt dargestellt werden.

Die den Schülern bekannten Grundlagen der Zahlentheorie<sup>58</sup> sollen zu Beginn der Unterrichtseinheit an Beispielen erörtert werden. Der *Hauptsatz der elementaren Zahlentheorie* wird als bekannt vorausgesetzt, ohne ihn weiter zu hinterfragen.

Martin Wagenschein äußerte 1980: „Der ebenso einfache wie geniale antike Beweis dafür, daß die Reihe der Primzahlen niemals abbrechen kann, gehört zu den wenigen wirklich unentbehrlichen Stücken des mathematischen Lehrgutes.“<sup>59</sup> Deshalb soll Euklids Beweis über die *Unendlichkeit der Primzahlmenge* in dieser Unterrichtseinheit nicht fehlen.<sup>60</sup> Eine tiefergehende Diskussion, wie sie etwa Glatfeld<sup>61</sup>, Scheu<sup>62</sup> oder Winter<sup>63</sup> vorschlagen, kann aus Zeitgründen nicht erfolgen.

Nach der Diskussion einfacher Primzahltests<sup>64</sup> und Primzahlsieben<sup>65</sup> wird der Kongruenzbegriff wegen der übersichtlicheren und einfacheren Formulierung mathematischer Aussagen eingeführt. Die Kongruenzrechnung soll im wesentlichen in Bezug zur Resteberechnung von Potenzen bei Division durch ganze Zahlen betrachtet werden, weil nur diese bei den Primzahltests eine Rolle spielt.

Die *Eulersche  $\varphi$ -Funktion* und der *Satz von Euler* werden nicht berührt. Der *Kleine Satz von Fermat* soll bewiesen werden, weil die hier zu behandelnden Primzahltests auf ihn gründen. Es wird ein Beweis des *Eulerschen Satzes* vereinfacht und auf den *Kleinen Satz von Fermat* angewendet.<sup>66</sup> Die Umkehrung des *Kleinen Satzes von Fermat* und die daraus resultierenden Primzahltests werden ausführlich behandelt. Dabei werden die *Carmichaelzahlen* und *Pseudoprimzahlen* definiert und detailliert mit Hilfe des Computeralgebra-Systems MuPAD untersucht. Bei einem Vergleich der Häufigkeiten von Primzahlen, *Carmichaelzahlen* und *Pseudoprimzahlen*, der für eine sehr grobe Abschätzung der Güte der Tests benötigt wird, soll vor allem mit Beispielen gearbeitet werden. Die *Primzahlverteilungsfunktion* wird in diesem Zusammenhang nicht definiert und der *Primzahlsatz*<sup>67</sup> nicht behandelt, weil die notwendigen Erkenntnisse über die Anzahl der Primzahlen bis zu einer gewissen Grenze mit der Angabe von einigen Primzahlanteilen ausreichend erscheint. Wegen dem Anspruch der zu behandelnden Primzahltests und der knapp bemessenen Unterrichtszeit sind hier Vereinfachungen notwendig. Die Erarbeitung des Pseudoprimzahltests soll durch das CAS MuPAD unterstützt werden. Je nachdem, wie gut der Einzelne die mathematischen Inhalte versteht und wie interessiert er sich zeigt, kann er Abwandlungen des Tests vornehmen, Primzahl Listen generieren und Vergleiche zwischen den einzelnen Tests durchführen.

<sup>57</sup> Dies wird natürlich dazu führen, dass die Tragweite der Bedeutung einzelner Sachverhalte nicht klar genug hervortritt und dass nicht alle Schüler diese Nebeninformationen vollständig erfassen.

<sup>58</sup> Vgl. Folie «Zahlentheorie: Grundlagen». ANHANG, 47.

<sup>59</sup> Winter 1991, 24.

<sup>60</sup> Wagenschein bemerkte außerdem: „Für die überhaupt dafür Empfänglichen ist das aktive Begreifen dieses souveränen Verfahrens ein unvergeßliches Erlebnis.“ (Winter 1991, S. 24) Dem wird der Unterricht nicht gerecht werden können, weil Primzahltests im Mittelpunkt des Unterrichtsversuches stehen. (Wagenschein selbst führte zu der Unendlichkeitsthematik der Primzahlen eine fünfständige Unterrichtsreihe durch (Vgl. Winter 1991, 24f.).)

<sup>61</sup> Vgl. Glatfeld 1993b, 5ff.

<sup>62</sup> Vgl. Scheu 1992, 119f.

<sup>63</sup> Vgl. Winter 1991, 22ff.

<sup>64</sup> Vgl. Arbeitsblatt «Erste Primzahltests». ANHANG, 55.

<sup>65</sup> *Sieb des Eratosthenes* und *Sieb des Ulam*.

<sup>66</sup> Vgl. Beweis des *Kleinen Satzes von Fermat*. ANHANG, 61. In Padberg 1999a, 69 findet man den Beweis des Eulerschen Satzes. Es wurde kein Beweis mittels der *vollständigen Induktion* gewählt, weil nicht alle Schüler diese Beweistechnik kennen.

<sup>67</sup> Primzahlsatz: Vgl. Remmert 1995, 72ff.

## 2. 4. Angestrebte Unterrichtsergebnisse

An dieser Stelle seien die grundlegenden Lernziele<sup>68</sup> der Unterrichtseinheit und nicht die angestrebten Unterrichtsergebnisse der einzelnen Stunden dargestellt.

- Die Schüler erfahren die Nützlichkeit und Reichweite mathematischer Grundlagenforschung. Die Anwendbarkeit der Mathematik für Aufgaben in Wissenschaft, Wirtschaft und Gesellschaft und ihre Bedeutung für die Zukunft<sup>69</sup> wird den Schülern an Beispielen bewusst.
- Die Schüler erhalten einen Einblick in einen Teil der Mathematikgeschichte, können die herausragenden Leistungen damaliger Mathematiker einschätzen und erkennen, dass die Mathematik eine sich stetig weiterentwickelnde Wissenschaft ist.
- Die Schüler wissen, dass in Computeralgebra-Systemen effiziente Algorithmen genutzt werden, die der mathematischen Forschung zu verdanken sind. Sie erkennen einerseits Grenzen der modernen Rechentechnik und auch der Mathematik, andererseits erfahren sie, wie die Mathematik ihre eigenen Limits für Anwendungen in der Wirtschaft intelligent auszunutzen versteht<sup>70</sup>.
- Die Schüler setzen sich mit vielfältigen Problemstellungen der Zahlentheorie auseinander. Sie kennen wichtige Begriffe und Regeln der Teilbarkeit sowie der Kongruenzrechnung und sind fähig, einzelne dieser Regeln anzuwenden. Sie erkennen die Bedeutung des *Kleinen Satzes von Fermat* für die Effizienz von Primzahltests und wissen um den Einfluss der *Carmichaelzahlen* und *Pseudoprimzahlen* bei diesen Tests. Sie können den Ablauf einzelner Primzahltests beschreiben.
- Die Schüler können den Computer sinnvoll und bedacht zur Lösung mathematischer Probleme einsetzen. Sie wissen, wie man in einem üblichen CAS programmiert und können einzelne Prozeduren in MuPAD implementieren und analysieren.

Bei der Erarbeitung der fachlichen Inhalte sollen fundamentale Denktätigkeiten und Denkhaltungen sowie geistige Grundtechniken, wie sie beispielsweise Zech aufführt<sup>71</sup>, gefördert werden.

Das fundamentale Ziel des Unterrichts ist es, die positive Grundeinstellung der Lernenden zur Mathematik zu fördern und Spaß im Umgang mit mathematischen Themen zu vermitteln.

---

<sup>68</sup> Der Begriff «Lernziele» soll die angestrebten Unterrichtsergebnisse umfassen. Die Ziele sollen trotz der eher strengen Formulierung, also ohne die Verwendung des Konjunktivs, so verstanden werden, dass der Unterricht einen Beitrag zum Erreichen dieser Absichten leisten soll. Die meisten Lernziele sind nicht so stark am fachlichen Inhalt ausgerichtet, weil dieser im Unterricht der folgenden Klassen wenig Bedeutung findet. Es soll vor allem die emotionale Seite der Lernenden angesprochen werden.

<sup>69</sup> Wittmann meint: „Die Mathematik ist weit über den technologischen Aspekt hinaus grundlegend für das Verständnis und die Erschließung der modernen Welt“ (Christmann 1980, 82f.).

<sup>70</sup> Beispielsweise nutzt das RSA-Verfahren (Verschlüsselungsverfahren von Rivest, Shamir und Adleman) die Unkenntnis um einen effizienten Algorithmus zur Faktorisierung von großen Zahlen und damit auch das Unwissen über das Verhalten der Primzahlen aus (Würde man ein einfache Gleichung kennen, die genau alle Primzahlen liefert, könnte man die Primfaktoren einer gegebenen Zahl relativ leicht finden, z.B. durch Multiplikationsversuche.).

<sup>71</sup> Vgl. Zech 1998, 54ff.

## 2. 5. Methodische Analyse

Die Beachtung der gegebenen Rahmenbedingungen führte zu einer Einteilung in eher lehrerzentrierte und handlungsorientierte Stunden. Die lehrerzentrierten Stunden tragen viele Elemente in sich, die über die Vermittlung der fachlichen Grundlagen hinausgehen und dem Erreichen der in Abschnitt 2.4. aufgeführten Ziele dienen. Es wird trotz des engen Zeitplans versucht, die Schüler aktiv werden zu lassen. Die vorherrschenden Methoden sind jedoch Lehrervortrag und Unterrichtsgespräch. Die handlungsorientierten Stunden, vor allem die am Computer, dienen nicht in erster Linie der Anwendung von Gelerntem, sondern sind unumgänglich für das Verständnis der fachlichen Inhalte; es handelt sich also um Erarbeitungsphasen.

Die Planung soll einer allgemeinbildenden Unterrichtskultur im Sinne von Heymann<sup>72</sup> möglichst gerecht werden.<sup>73</sup> Dieses Ziel wird in den theoretisch ausgelegten Stunden, nicht zuletzt wegen der stofflichen Fülle, leider nicht durchgängig erreichbar sein, in den handlungsorientierten Stunden jedoch einen größeren Stellenwert einnehmen können.

### 2. 5. 1. Motivation

Die Konzeption einer Unterrichtseinheit muss berücksichtigen, dass die Schüler regelmäßig motiviert werden. Deshalb wird die Frage «Warum beschäftigte und beschäftigt man sich mit Primzahlen?» in der ersten Doppelstunde nicht erschöpfend beantwortet, sondern taucht u.a. in der siebenten Stunde wieder auf.<sup>74</sup> Neben den Anreizen einzelner Aufgaben und Problemstellungen sind größere Motivationsschübe in der ersten, dritten, sechsten und siebenten Stunde geplant.<sup>75</sup>

Beachtet werden muss der zumeist positive Einfluss, den der Rechner auf viele Lernende bisher ausgeübt hat. Vier der geplanten Unterrichtsstunden sind für die Arbeit mit MuPAD reserviert.<sup>76</sup> In der dritten Stunde wird der CAS – Einsatz u.a. eingeschoben, um die Theorie etwas aufzulockern. Innerhalb einiger Einzelstunden soll der Computer zu Präsentationszwecken genutzt werden. Die Aktualität der gewählten Inhalte des Internets und die Einsicht, dass jeder der Schüler auf diese Informationen Zugriff hat, sollen zur Auseinandersetzung mit der Thematik anspornen.

### 2. 5. 2. Vollständigkeit und Ausführlichkeit

Die Zeitplanung gestattet sowohl bei den in der ersten Stunde zu behandelnden Teilbarkeitseigenschaften, die hier aufbauend auf das letzte MuPAD-Arbeitsblatt<sup>77</sup> in Verbindung mit der Sicherung des Ausgangsniveaus<sup>78</sup> erarbeitet werden, als auch bei den Rechenregeln für Kongruenzen keine Vollständigkeit und Ausführlichkeit. Die Schüler sollen lediglich ein Gefühl für den Umgang mit den einzelnen Regeln gewinnen. Wie in Abschnitt 2.1. beschrieben, sind die Jugendlichen in der Lage, ihr eigenes Wissen kritisch einzuschätzen und selbständig Lücken zu schließen. Zur Unterstützung wird den Lernenden jeweils eine ausführliche Zusammenfassung in Form eines Arbeitsblattes zur Verfügung gestellt.

### 2. 5. 3. Logische Abfolge der Inhalte – Einführung des Kongruenzbegriffs

Der Unterricht ist so organisiert, dass einzelne Teile möglichst fließend ineinander übergehen. Schwierigkeiten bereitet hier vor allem der Kongruenzbegriff. Der Sinn der Einführung kann den Lernenden nahe gebracht werden, indem man auf den Nutzen bei effizienteren Primzahltests verweist.<sup>79</sup> Doch erscheint der Übergang vom Primzahl- zum Kongruenzbegriff mit dem Hinweis «Wir müssen hier noch schnell etwas kennen lernen, was uns später nutzt.» etwas holprig. Es besteht die Möglichkeit, direkt von der Teilbarkeit zum Kongruenzbegriff überzugehen. Dann würde man aber zu viel Zeit benötigen, bis man zu den Primzahltests kommt und die Lernenden am Anfang mit zu viel Theorie die Freude auf die weiteren Stunden verderben. Einen Ausweg zeigt

<sup>72</sup> Hans Werner Heymann.

<sup>73</sup> Vgl. Heymann 1996, 262ff.

<sup>74</sup> In der ersten Stunde soll vor allem die innermathematische Bedeutung der Primzahlen von den Lernenden aus dem Hauptsatz der elementaren Zahlentheorie abgeleitet werden. Die Anwendungen innerhalb der Kryptographie werden in der siebenten Stunde thematisiert.

<sup>75</sup> In der dritten Stunde wird die «Primzahljagd» vorgestellt. Die sechste Stunde offenbart Vorteile der Kongruenzrechnung.

<sup>76</sup> Inwieweit die relativ lange Arbeit am Rechner noch motiviert, wird der Unterrichtsversuch ergeben.

<sup>77</sup> Vgl. MuPAD-Arbeitsblatt 8. ANHANG, 46.

<sup>78</sup> Es werden neben den Teilbarkeitseigenschaften grundlegende Definitionen und Sätze im Unterrichtsgespräch wiederholt (Vgl. Arbeitsblatt «Zahlentheorie: Grundlagen». ANHANG, 47).

<sup>79</sup> Es sei an dieser Stelle angemerkt, dass man die Primzahltests, auch den *Kleinen Satz von Fermat*, durchaus formulieren kann, ohne den Kongruenzbegriff zu verwenden. Die Übersicht der mathematischen Formulierungen würde jedoch sehr leiden. Außerdem sind die zu nutzenden MuPAD-Befehle sehr eng mit dem Kongruenzbegriff verbunden, so dass eine Erklärung spätestens hier notwendig sein würde.

das gewählte Vorgehen. Das *Sieb des Ulam* bietet die Chance, Erkenntnisse über Primzahlen zu sammeln.<sup>80</sup> Untersucht man das Sieb genauer, stellt man fest, dass die Zahlen in einer Spalte jeweils eins gemeinsam haben: Sie lassen bei Division durch 6 denselben Rest. Der Übergang zum Kongruenzbegriff ist geschaffen.

#### 2. 5. 4. Lehrervortrag und Unterrichtsgespräch

Bei der Erarbeitung einiger interessanter Sachverhalte muss manchmal trotz kritischer Auswahl der Inhalte auf weitestgehend selbständige Schülerhandlungen verzichtet werden. Vor allem die erste und dritte Doppelstunde werden vom Lehrervortrag und Unterrichtsgespräch geprägt sein.<sup>81</sup> Beide Methoden stellen hohe Anforderungen an die Jugendlichen. Obwohl die meisten Schüler dieser Klasse in der Lage sind, sich über einen längeren Zeitraum zu konzentrieren und die Mathematikstunden am Anfang des Tages erteilt werden, wird die dritte Doppelstunde einige überfordern. In den ersten beiden Stunden der Unterrichtseinheit ist die Stofffülle zwar groß, aber bis auf Euklids Beweis verlangen die fachlichen Inhalte nicht durchgängig kognitive Höchstleistungen. Für die Inhalte der dritten Doppelstunde wäre es wünschenswert, zwei Stunden getrennt voneinander unterrichten zu können. Dies erscheint jedoch aufgrund der gegebenen Bedingungen nicht möglich. Die Lehrervorträge und Unterrichtsgespräche werden durch viele Beispiele und Medien<sup>82</sup> unterstützt. Den Lernenden sollen außerdem Gelegenheiten zum «Durchatmen» geschaffen werden, indem die logischen Ableitungen durch philosophische und historische Betrachtungen<sup>83</sup> und witzige Bemerkungen<sup>84</sup> aufgelockert werden.

#### 2. 5. 5. Beweise

Beweise spielen in der Mathematik eine besondere Rolle, sie verdeutlichen die Exaktheit dieser Wissenschaft.<sup>85</sup> Im Unterrichtsverlauf sind zwei größere Beweise eingeplant, die beide im Lehrervortrag dargeboten werden: Euklids Beweis über die *Unendlichkeit der Primzahlmenge* und der Beweis des *Kleinen Satzes von Fermat*.<sup>86</sup> Der Beweis von Euklid wird einerseits wegen der Verbindung zur Geschichte gewählt, andererseits weil er verdeutlicht, wie raffiniert große Mathematiker mit ihrer Unwissenheit umzugehen wissen<sup>87</sup>. Der hier geführte Beweis des *Kleinen Satzes von Fermat*<sup>88</sup>, der nach der Betrachtung von repräsentativen Beispielen erfolgen soll<sup>89</sup>, erscheint für die Lernenden sicher etwas konstruiert und viele werden Schwierigkeiten haben, den logischen Gedankengängen zu folgen. Man kann mit einem derart schwierigen, für den Schulalltag nicht alltäglichen Beweis, den Schülern dieser Klasse jedoch näher bringen, dass Mathematiker nicht wie selbstverständlich neue Erkenntnisse ermitteln. Die Mathematik ist eine Wissenschaft, in der erst die gründliche Auseinandersetzung mit mathematischen Grundlagen neue Erfahrungen ermöglicht.

Um Zeiteinbußen zu vermeiden, sollen weitere kleinere Nachweise von Schülern in freiwilliger zusätzlicher Arbeit oder vom Lehrenden geführt werden.

#### 2. 5. 6. Das Computeralgebra-System MuPAD

Für den Unterrichtsversuch wurde das CAS MuPAD ausgewählt. MuPAD ist nicht speziell für die Schuleinsatz entwickelt und es gibt z.B. im Vergleich zu DERIVE kaum Einsatzerfahrungen innerhalb des Unterrichts. Die Arbeit erfolgt befehlsorientiert, deshalb muss eine Einarbeitungsphase berücksichtigt werden.<sup>90</sup> MuPAD stellt ein mächtiges Werkzeug dar: „Man kann davon ausgehen, dass kein Bereich aus der Mathematik nicht in vielfältiger

<sup>80</sup> Thematisiert werden u.a. Primzahlzwillinge und Primzahlzdrillinge.

<sup>81</sup> Euklids Beweis zur *Unendlichkeit der Primzahlmenge*, der Test 2 und der *Kleine Satz von Fermat* werden vom Unterrichtenden bewiesen. Das Unterrichtsgespräch zur Umkehrung des *Kleinen Satzes von Fermat* wird vom Lehrenden stark gesteuert.

<sup>82</sup> Hier sind vor allem Folien bzw. Kopien zu nennen, die möglichst aktuelles Datenmaterial enthalten. Eine Präsentation mit dem Computer ist auf Grund der räumlichen und materiellen Bedingungen in den Doppelstunden nicht möglich.

<sup>83</sup> Es wird u.a. diskutiert, wie, warum und mit welchem Erfolg sich Mathematiker mit Primzahlen beschäftigten.

<sup>84</sup> Vgl. Folie «Alle ungeraden Zahlen größer Eins sind Primzahlen». ANHANG, 50.

<sup>85</sup> K. Urbanik: „In der Mathematik gibt es keine Autoritäten. Das einzige Argument für die Wahrheit ist der Beweis.“ (<http://www.pirabel.de/zitate.htm>).

<sup>86</sup> Der Beweis von Euklid wird auf Grund des eng bemessenen zeitlichen Rahmens, der Beweis des *Kleinen Satzes von Fermat* wegen des hohen fachlichen Anspruchs vom Lehrenden geführt.

<sup>87</sup> Euklid tut nur das Nötigste: Er gibt eine Gleichung zur Konstruktion einer neuen Primzahl an, die weder alle Primzahlen erzeugt, noch die neue Primzahl direkt erkennen lässt. Die Unwissenheit um eine Primzahlformel, die alle Primzahlen und nur diese beschreibt, umgeht er damit sehr geschickt.

<sup>88</sup> Vgl. Beweis des *Kleinen Satzes von Fermat*. ANHANG, 61.

<sup>89</sup> Vgl. Arbeitsblatt «Kleiner Satz von Fermat». ANHANG, 60.

<sup>90</sup> Die Lernenden besitzen bereits Vorkenntnisse (Vgl. Abschnitt 2.1.). Es wird sich im Laufe des Unterrichts zeigen, wie intensiv sich die freiwillige häusliche Arbeit mit MuPAD gestaltete.

Form innerhalb des Systems verfügbar ist.<sup>91</sup> Das prozedurale Programmierkonzept ist für die Arbeit innerhalb der geplanten Unterrichtseinheit von Vorteil. Außerdem ist eine Light-Version mit eingeschränktem Bedienkomfort kostenfrei erhältlich. Für die Dauer eines Projekts wird die Vollversion MuPAD-Pro ebenfalls kostenlos angeboten. So kann MuPAD während des Unterrichtsversuches ständig, auch zu Hause, von den Lernenden genutzt werden. Die freie Verfügbarkeit und die Leistungsfähigkeit sprechen für die Verwendung dieses CAS innerhalb des Unterrichtsversuches.

### 2. 5. 7. Konzeption der vom CAS – Einsatz geprägten Stunden

Die Planung der Stunden, in denen die Lernenden MuPAD nutzen, muss u.a. das unterschiedliche Arbeitstempo der Schüler berücksichtigen. Sie erhalten deshalb die Aufträge auf einem Arbeitsblatt. Am Anfang der Stunde wird besprochen, welche Aufgaben bewältigt werden müssen und welche zusätzlich bearbeitet werden können. Die Arbeitsblätter sind so angelegt, dass neben den Aufgaben auch fachliche Grundlagen und Hinweise zur Umsetzung aufgeführt sind. Die Ratschläge werden in den ersten Stunden detaillierter Natur sein, in den letzten Stunden werden weniger Hinweise gegeben. Der Anspruch an die Lernenden wird schrittweise erhöht.

Der Ergebnissicherung sollte innerhalb dieser Stunden besondere Beachtung zukommen. Die geschriebenen Prozeduren können die Schüler auf Diskette speichern und zu Hause weiter bearbeiten, geforderte Antworten müssen schriftlich auf dem Arbeitsblatt dargelegt werden<sup>92</sup>. Außerdem erfolgen am Ende einer jeden Stunde mündliche Zusammenfassungen und eine umfassende Diskussion der Ergebnisse. Nach abschließender Besprechung werden den Lernenden die Prozeduren in schriftlicher Form zur Verfügung gestellt.

Der Lehrende nimmt eine beratende Funktion ein, gibt individuelle Hilfen und fördert unterschiedliche Lösungswege sowie deren Besprechung.

Leider stehen nicht genügend Rechner zur Verfügung, so dass jeweils zwei Schüler an einem Computer arbeiten müssen. Der Unterrichtende muss darauf achten, dass die Partner sich bei der Arbeit abwechseln.

Es ist möglich, dass große Unterschiede im Leistungsvermögen der Schüler auftreten. Differenzierte Hilfen müssen das Grundniveau sichern. Weil jeder das CAS zu Hause nutzen kann, ist es möglich, einzelne Aufgaben<sup>93</sup> in die Hausaufgabe zu verlagern.

### 2. 5. 8. Primzahltests und ihre Erarbeitung mit einem CAS

Primzahltests bilden den Schwerpunkt der Unterrichtsthematik. Erste Tests<sup>94</sup> lassen sich direkt aus der Primzahldefinition ableiten. Diese und auch die Pseudoprimzahltests sollen in einer ersten Phase im Unterrichtsgespräch erarbeitet werden, damit die Umsetzung im CAS nicht in ein Vorgehen nach Versuch und Irrtum «ausartet». Ein tieferes Verständnis für die Primzahltests sollen die Lernenden erhalten, wenn sie sich bei der Programmierung der Tests mit den Voraussetzungen für einzelne Entscheidungen eingehend auseinandersetzen. Die dritte Stunde soll u.a. noch einer Einarbeitung in das prozedurale Konzept dienen. In den Stunden neun, zehn und elf soll das CAS die Erarbeitung der einzelnen Prüfbedingungen unterstützen und den Schülern die Eigenheiten der Primzahltests näher bringen. Die Lernenden sollen das Wesen der *Carmichaelzahlen* und *Pseudoprimzahlen* verstehen, indem sie ihre Besonderheiten und ansatzweise die Häufigkeit ihres Auftretens untersuchen. Die Lernenden sollen selbst Primzahltests erstellen und Beispielrechnungen durchführen, die neue Erkenntnisse hervorbringen. Außerdem sollen sie ein Gefühl für die Größe der heutzutage in der Forschung interessanten Primzahlen bekommen. Der Einsatz eines Computeralgebra-Systems innerhalb dieser Thematik ist wichtig. Ohne einen solchen sind kaum repräsentative Rechnungen möglich.<sup>95</sup> Ein Vergleich der einzelnen Primzahltests und somit eine Legitimation des Themas kann erst innerhalb dieser Stunden erfolgen. Hier können und müssen die Lernenden erfahren, dass die eingehende Beschäftigung mit einer Thematik fulminante Ergebnisse hervorbringen kann, dass sich die Auseinandersetzung mit mathematischen Sachverhalten lohnt.

<sup>91</sup> Sodis-Datenbank: MuPAD – The Open Computer Algebra System: LSW M 850. 1999. In: <http://www.sodis.de/>.

<sup>92</sup> Zu den meisten Aufgaben sind Fragen formuliert, deren Beantwortung eine Überprüfung der Prozeduren ermöglicht und für die Lernenden neue Erkenntnisse sichtbar werden lässt.

<sup>93</sup> Dies können leichte Aufgaben oder freiwillig zu lösende Zusatzaufgaben sein. Außerdem besteht die Möglichkeit, Prozeduren nachzuarbeiten.

<sup>94</sup> Vgl. Arbeitsblatt «Erste Primzahltests». ANHANG, 55.

<sup>95</sup> Beispielsweise dauert die Überprüfung der zweiten *Carmichaelzahl* auf die Carmichaieleigenschaft mit einem aktuellen PC in MuPAD (ohne die Verwendung von *powermod*) rund 5 Sekunden.

### 2. 5. 9. Kontrolle der Lernergebnisse

Am Ende der Unterrichtseinheit erfolgt eine Lernerfolgskontrolle.<sup>96</sup> Die Bewertung soll in Form einer Note erfolgen, um eine Gesamtbeurteilung zum Abschluss des Schuljahres zu ermöglichen. Sie muss außerdem eine von der Schulleitung geforderte schärfere Ahndung von Formverstößen berücksichtigen.

Neben der Überprüfung von Gelerntem und dem Anwendung von Algorithmen werden einige Fragen auf das Verständnis der behandelten Primzahltests abzielen. Eine praktische Tätigkeit am Rechner ist auf Grund der begrenzten Rechneranzahl nicht geplant.<sup>97</sup> Das Anforderungsniveau der Kontrolle soll nicht zu hoch angesiedelt sein, weil das zu behandelnde Thema zusätzlich zum Lehrplan unterrichtet wird und wenige Erfahrungen in der Behandlung von Primzahltests im Unterricht einer 11. Klasse die Abschätzung möglicher Probleme erschwert.

---

<sup>96</sup> Vgl. Aufgabenblatt «Kontrolle: Primzahlen und Primzahltests». ANHANG, 72.

<sup>97</sup> Die Unruhe beim Wechseln der Plätze würde die Arbeitsatmosphäre stark beeinträchtigen.

## 2. 6. Verlaufsplanung

Neben den dargelegten Analysen berücksichtigt die Planung die räumlichen Gegebenheiten und die zur Verfügung stehende Unterrichtszeit von 12 Stunden. Der Computerraum kann jeweils nur in der Einzelstunde genutzt werden. In der vierten Doppelstunde war es möglich, in den Informatikraum zu wechseln.

**Tabelle 1: Übersicht über die Stundeneinteilung**

Stunde	Inhalt	Bemerkungen zur Durchführung
1./2.	Zahlentheoretische Grundlagen	EA/UG: Wiederholung bekannter Begriffe; Diskussion von Teilbarkeitseigenschaften; Zusammenfassung für die Lernenden <sup>98</sup>
	Bedeutung der Primzahlen	UG: Warum beschäftigte und beschäftigt man sich mit Primzahlen? <sup>99</sup>
	<i>Unendlichkeit der Primzahlmenge</i>	LV/UG: Verteilung der Primzahlen <sup>100</sup> ; Vermutungen der Lernenden LV: Beweis des Satzes über die <i>Unendlichkeit der Primzahlmenge</i> nach Euklid; Besonderheiten des Beweises
	Erste Primzahltests	UG: Aus der Primzahldefinition folgt Test 1 <sup>101</sup> . UG: Muss man alle Zahlen $a \in \mathbb{N}$ mit $1 < a < n$ testen? $\Rightarrow$ Test 2 <sup>102</sup> LV: Beweis von Test 2
	Suche nach einer Primzahlfolge	UG: Ungerade Zahlen <sup>103</sup> LV: Definition der <i>Fermatschen</i> und <i>Mersenneschen Zahlen</i> HA: Überprüfung der <i>Fermatschen</i> und <i>Mersenneschen Zahlen</i> auf die Primzahleigenschaft <sup>104</sup>
3.	Große Primzahlen und ihre Entdeckungen	UG: Vergleich der Hausaufgabe LV: <i>Fermatsche</i> und <i>Mersennesche Zahlen</i> ; Primzahlsuche im Internet <sup>105</sup>
	Erste Primzahltests in MuPAD	PA: Tests 1 und 2 in MuPAD <sup>106</sup> HA: Vervollständigen der Primzahltests in MuPAD
4./5.	<i>Sieb des Eratosthenes</i> und <i>Sieb des Ulam</i>	EA: Arbeitsblatt <sup>107</sup> UG: Vergleich der Siebe; Folgerungen aus dem <i>Sieb des Ulam</i>
	Kongruenzen; Kongruenzrechnung	UG: Gemeinsamkeiten der Zahlen in einer Spalte beim <i>Sieb des Ulam</i> LV: Kongruenzbegriff EA/UG: Ausgewählte Rechenregeln für Kongruenzen anhand von Beispielen; Zusammenfassung für die Lernenden <sup>108</sup>
6.	Kongruenzrechnung	LV, EA: Reste berechnen UG: Vorteile der Kongruenzrechnung HA: Reste berechnen

<sup>98</sup> Vgl. Arbeitsblatt «Zahlentheorie: Grundlagen». ANHANG, 47.

<sup>99</sup> Vgl. Folie «Primzahlen – damals und heute». ANHANG, 48.

<sup>100</sup> Vgl. Folie «Wie viele Primzahlen gibt es?». ANHANG, 49.

<sup>101</sup> Test 1:  $n$  Primzahl  $\Leftrightarrow$  Für alle  $a \in \mathbb{N}$  mit  $1 < a < n$  gilt:  $\text{ggT}(a, n) = 1$ .

<sup>102</sup> Test 2:  $n$  Primzahl  $\Leftrightarrow$  Für alle  $a \in \mathbb{N}$  mit  $1 < a \leq \lfloor \sqrt{n} \rfloor$  gilt:  $\text{ggT}(a, n) = 1$ .

<sup>103</sup> Vgl. Folie «Alle ungeraden Zahlen größer Eins sind Primzahlen». ANHANG, 50.

<sup>104</sup> Vgl. Arbeitsblatt «Fermatsche und Mersennesche Zahlen». ANHANG, 51.

<sup>105</sup> Vgl. Folie «Fermatsche Zahlen». ANHANG, 52; Folie «Mersennesche Zahlen». ANHANG, 53f.

<sup>106</sup> Vgl. Arbeitsblatt «Erste Primzahltests». ANHANG, 55.

<sup>107</sup> Vgl. Arbeitsblatt «Sieb des Eratosthenes und Sieb des Ulam». ANHANG, 57.

<sup>108</sup> Vgl. Arbeitsblatt «Rechenregeln für Kongruenzen». ANHANG, 58.

**Fortsetzung von Tabelle 1**

Stunde	Inhalt	Bemerkungen zur Durchführung
7./8.	Faktorisierung großer Zahlen	LV: Warum benötigt man große Primzahlen? <sup>109</sup>
	<i>Kleiner Satz von Fermat</i> und sein Beweis	LV: <i>Kleiner Satz von Fermat</i> EA/UG: Erarbeitung des Satzes <sup>110</sup> LV: Beweis des Satzes <sup>111</sup>
	Umkehrung des <i>Kleinen Satzes von Fermat</i>	UG: Umkehrung des Satzes LV: Vorstellung der Tests 3 <sup>112</sup> und 4 <sup>113</sup> ; Definition der <i>Carmichaelzahlen</i> und <i>Pseudoprimzahlen</i> ; EA/UG: Güte der Tests und praktische Tauglichkeit <sup>114</sup>
9.	<i>Carmichaelzahlen</i> und <i>Pseudoprimzahlen</i> in MuPAD	LV: <i>Carmichaelzahlen</i> , <i>Pseudoprimzahlen</i> und Primzahltests <sup>115</sup> PA: Arbeitsblatt <sup>116</sup>
10./11.	Primzahltests in MuPAD	PA: Arbeitsblatt <sup>117</sup> UG: Primzahltests <sup>118</sup>
12.	Kontrolle	EA: Aufgabenblatt <sup>119</sup>

<sup>109</sup> Vgl. Folie «Faktorisierung großer Zahlen». ANHANG, 59.

<sup>110</sup> Vgl. Arbeitsblatt «Kleiner Satz von Fermat». ANHANG, 60.

<sup>111</sup> Vgl. Beweis des *Kleinen Satzes von Fermat*. ANHANG, 61.

<sup>112</sup> Test 3: Gilt für alle  $a \in \mathbb{N}$  ( $1 < a < n$ ) mit  $\text{ggT}(a, n) = 1$ :  $a^{n-1} \equiv 1 \pmod{n} \Rightarrow n$  ist Primzahl.

<sup>113</sup> Test 4: Gilt für einige  $a \in \mathbb{N}$  mit  $1 < a < n - 1$ :  $\text{ggT}(a, n) = 1$  und  $a^{n-1} \equiv 1 \pmod{n} \Rightarrow n$  ist Primzahl.

<sup>114</sup> Vgl. Folie «Carmichaelzahlen». ANHANG, 63; Folie «Pseudoprimzahlen». ANHANG, 64.

<sup>115</sup> Vgl. Arbeitsblatt «Übersicht über Primzahltests». ANHANG, 65.

<sup>116</sup> Vgl. Arbeitsblatt «Carmichaelzahlen und Pseudoprimzahlen». ANHANG, 66f.

<sup>117</sup> Vgl. Arbeitsblatt «Primzahltests 3 und 4». ANHANG, 68f.

<sup>118</sup> Vgl. Arbeitsblatt «Carmichaelzahlen und Pseudoprimzahlen in MuPAD». ANHANG, 70; Arbeitsblatt «Primzahltests in MuPAD». ANHANG, 71.

<sup>119</sup> Vgl. Aufgabenblatt «Kontrolle: Primzahlen und Primzahltests». ANHANG, 72.

### 3. 3. Bewertung der Planung und Ansätze zur Verbesserung

In Abschnitt 3.1. wurden bereits die während des Unterrichts aufgetretenen Probleme dargelegt. Hier sollen nun wesentliche Aspekte der Planung beurteilt und Verbesserungsansätze für eine erneute Durchführung gegeben werden.

Die Konzeption einer Unterrichtseinheit hängt grundsätzlich von den zu unterrichtenden Schülern, u.a. von ihren Neigungen, Fähigkeiten und Vorkenntnissen sowie von der räumlich-zeitlichen Konstellation ab. Der Unterrichtsgang ist unter anderen als den hier gegebenen Bedingungen nicht ohne weiteres übertragbar, deshalb sollen sich die angebotenen Verbesserungsvorschläge auf diese beziehen.

Der sachlogische Aufbau des Themas ist trotz der vorgenommenen Kürzungen und Vereinfachungen zufriedenstellend. Schwierigkeiten bereitete der enge Zeitplan in den eher theoretisch ausgerichteten Doppelstunden. So konnten der Satz über die *Unendlichkeit der Primzahlmenge* und sein Beweis nicht genügend diskutiert werden. Man ist geneigt, zumindest den Beweis gänzlich aus der Unterrichtseinheit zu streichen. Dazu muss man die Frage beantworten, wie wichtig seine Behandlung für die Thematik ist. Der Beweis lässt neben der genialen Beweisführung u.a. erahnen, wie unregelmäßig sich die Primzahlen verhalten. Eine ausführlichere Betrachtung der Primzahlfolge, mit Berücksichtigung weiterer Formeln, die viele Primzahlen generieren, könnte diese Erkenntnisse jedoch auch hervorbringen. Ein solches Vorgehen bietet außerdem den Vorteil, dass die Lernenden mehr aktiv werden können.

An der Konzeption der zweiten Doppelstunde und der sechsten Stunde sollte man trotz der angesprochenen Motivationsprobleme einiger Schüler festhalten, weil nicht die Planung für diese Schwierigkeiten verantwortlich zu sein scheint. Natürlich wäre es wünschenswert, die Übung zur Kongruenzrechnung, z.B. mit Herleitungen von Teilbarkeitsregeln, auszuweiten. Die relativ kurze Behandlung reicht aber aus, um diesen Lernenden die für die Primzahltests notwendigen Grundfertigkeiten nahe zu bringen. Beachtung sollte eine Wiederholung der Potenzgesetze finden.

Die größten Probleme beinhaltete die dritte Doppelstunde, weil der dauerhaft hohe fachliche Anspruch und die gewählten Methoden über die Zeit von zwei Unterrichtsstunden die Lernenden überforderten. Die Inhalte sollten jedoch nicht verändert werden. Der Beweis des *Kleinen Satzes von Fermat* wurde der Grenze des Leistungsvermögens einiger mathematisch begabter Schüler dieser Klasse gerecht, denn auch diese Lernende müssen im Mathematikunterricht stark gefordert werden und Gelegenheit bekommen, ihre Limits auszuloten. Eine dauernde Überlastung kann man vermeiden, indem man die sechste und siebente Stunde tauscht. Die Einführung der Rechenregeln für Kongruenzen reicht aus, um den *Kleinen Satz von Fermat* formulieren und beweisen zu können. Der Beweis wird an einigen Stellen etwas schwieriger zu verstehen sein, doch können einzelne Beweisschritte nun mit Hilfe des CAS – Einsatzes durch Beispielrechnungen unterstützt werden. Der Stundentausch wird in erster Linie dem Verständnis des Satzes zu Gute kommen, weil sich die Restberechnungen im Nachhinein nun direkt auf den *Kleinen Satz von Fermat* beziehen können und die Vorteile des Satzes so stärker in den Vordergrund treten. Die Übungsstunde kann dementsprechend besser motiviert werden und die von Lehrervortrag und Unterrichtsgespräch geprägten Stunden werden voneinander getrennt.

In Anbetracht der selbständigen Einarbeitung in MuPAD traten relativ wenige Problem bei der Arbeit mit dem CAS auf. Man sollte trotzdem am Anfang möglichst sehr sensibel in die CAS – Programmierung einführen, einzelne Prozeduren oft besprechen und differenzierte Aufgabenstellungen geben. Die Planung der vom CAS geprägten Stunden, insbesondere auch die Arbeitsblätter, stellten sich als brauchbar heraus, jedoch sollte das Aufgabenblatt zu den Tests 1 und 2 eine gründliche Analyse einer Prozedur verlangen. Die Eigenheit, dass ein Primzahltest zuerst versucht, eine zusammengesetzte Zahl nachzuweisen und erst am Ende «Primzahl» ausgibt, müsste im theoretischen Teil des Unterrichts mehr thematisiert werden.

Die ursprüngliche Konzeption vermochte die Schüler für die Thematik aufzuschließen der nicht erwarteten Veränderung der Klassensituation konnte sie jedoch nicht gerecht werden. Die neue Planung sollte trotzdem nicht von Notendruck, ständigen Kontrollen u.Ä. geprägt sein, weil ein solches Vorgehen dem Erreichen der meisten Unterrichtsziele entgegenwirkt und der Unterricht in dieser Klasse viel von seiner «Spritzigkeit» verlieren würde. Deshalb wird sich auch die verbesserte Planung einer solchen unvorhersehbaren Entwicklung nicht stellen können.<sup>120</sup>

Die beschriebenen Ansätze zur Verbesserung der Unterrichtseinheit führen zu einer neuen Verlaufsplanung.<sup>121</sup>

<sup>120</sup> Wahrscheinlich wurden die Lernenden durch die Häufung von Klausuren zum Ende des Schuljahres überlastet. Es ist damit zu rechnen, dass sich die Arbeitsmoral der Klasse wieder bessern wird.

<sup>121</sup> Vgl. Neue Verlaufsplanung. ANHANG, 82.

### 3. 4. Zahlentheorie in der gymnasialen Oberstufe

Die Zahlentheorie bietet vielfältige Möglichkeiten für eine Behandlung innerhalb des Unterrichts der höheren Klassen, das gewählte Thema ist dabei eines von vielen. Einige Inhalte der elementaren Zahlentheorie tragen einen niedrigeren fachlichen Anspruch als die betrachteten Primzahltests, die jedoch den Vorteil haben, dass sie in der heutigen Zeit eine wichtige Rolle im Zusammenhang zur Kryptographie spielen. Der Unterricht bleibt nicht nur bei einer historischen Betrachtung von großen mathematischen Entdeckungen stehen, sondern schafft Verbindungen zur aktuellen Forschung.

Es ist jedoch kaum möglich, alle notwendigen mathematischen Voraussetzungen mit der gewünschten Akkuratess zu behandeln, das Versäumnis des Mathematikunterrichts, sich nicht stetig zahlentheoretischen Problemen zu widmen, nachzuholen. Die notwendigen Voraussetzungen der behandelten Primzahltests, vor allem der *Kleine Satz von Fermat*, stellen sehr hohe Anforderungen an die Schüler einer 11. Klasse. Die schnelle Aufeinanderfolge und Abarbeitung der Fülle von Grundlagen haben Verständnisschwierigkeiten zur Folge, Handlungen werden nicht genügend verinnerlicht. Innerhalb einer Klasse mit «normalem» Leistungsniveau müsste die systematische Erarbeitung solcher Grundlagen mehr Raum einnehmen, so dass der hier gegebene Zeitrahmen nicht ausreichend erscheint. Die äußeren Rahmenbedingungen, vor allem die Lernenden dieser Klasse, die den Willen und die Fähigkeit besitzen<sup>122</sup>, sich höheren mathematischen Herausforderungen zu stellen, machen die Behandlung dieses Teilgebiets der elementaren Zahlentheorie im vorgegebenem Zeitrahmen möglich. Das gewählte Thema ist demnach nur bedingt für den Unterricht in einer 11. Klasse geeignet.

In einem Leistungskurs der Jahrgangsstufe 12 oder 13 würde der Unterricht aufgrund der gesteigerten Denk- und Argumentationsfähigkeit sicherlich besser durchgeführt werden können. Außerdem ist eine Bearbeitung des Themas innerhalb einer Projektwoche bzw. eines Förderunterrichtes für mathematisch begabte Schüler denkbar.

---

<sup>122</sup> Es sei angemerkt, dass trotz der erwähnten Verringerung der Lernbereitschaft einige Schüler während der gesamten Unterrichtseinheit engagiert und konzentriert am Unterrichtsgeschehen beteiligt waren.

### 3. 5. Der CAS – Einsatz im Unterricht

Die Behandlung der Primzahltests ist ohne die Verwendung eines CAS, ohne die Möglichkeit, die Tests in der Praxis zu erproben und zu untersuchen, kaum denkbar. Die von der Theorie etwas überforderten Schüler konnten erst in der praktischen Arbeit das Wesen der *Carmichaelzahlen* und der *Pseudoprimzahlen* erfassen, die Zahlenarten unterscheiden, die Laufzeiten verschiedener Primzahltests vergleichen und damit Einblicke in die Notwendigkeit effizienter Algorithmen gewinnen, die Sicherheit der Pseudoprimzahltests abschätzen u.s.w.

Fruchtbar für die Schüler, sicher auch anstrengend für den Unterrichtenden, erwiesen sich die spezifischen Hilfestellungen, die der Lehrende geben konnte. Einzelne Gespräche ermöglichen neben der effizienten Beseitigung individueller fachlicher Probleme, die Schüler für die Mathematik aufzuschließen. So konnten während der Unterrichtseinheit einige Lernende zur tieferen Auseinandersetzung mit der Thematik außerhalb des Unterrichts angespornt werden.<sup>123</sup>

Der Einsatz eines Computeralgebra-Systems bringt einige Probleme mit sich.

Es gibt Lernende, für die der Rechner eine große Hemmschwelle darstellt. Ständiger Computereinsatz kann bei diesen zur Frustration führen. Das behutsame Heranführen dieser Schüler an die Arbeit mit dem PC sollte auch Aufgabe des Mathematikunterrichts sein.

Die Sprache des verwendeten CAS stellt an Schüler, die das Programmieren nicht gewohnt sind, zusätzliche Anforderungen. Viele benötigen einige Zeit, sich an die gegebene Programmiersprache und an das Programmieren selbst zu gewöhnen. Innerhalb dieser Klasse waren die Voraussetzungen, nicht zuletzt wegen der freiwilligen selbständigen Einarbeitung in MuPAD, günstig, so dass hier kaum Schwierigkeiten zu verzeichnen waren. Dies wird nicht ohne weiteres auf andere Schülergruppen übertragbar sein.

Selbst ein befehlsorientiertes CAS wie MuPAD spricht die Lernenden emotional an, verleitet demzufolge auch zum Spielen.<sup>124</sup> Der glückliche Umstand, dass jeder MuPAD auch zu Hause verwenden konnte, begünstigte die Einhaltung der vom Lehrenden aufgestellten Regel «Tests über drei Minuten werden nicht in der Unterrichtszeit durchgeführt.»

Die Schüler arbeiten meist zu zweit an einem Rechner. Deshalb sind die Kontrollmöglichkeiten des Unterrichtenden eingeschränkt. Man sollte darin nicht nur einen Nachteil sehen, weil sich die Chance bietet, die Eigenverantwortlichkeit der Jugendlichen zu fördern. Die Schüler dieser Klasse unterstützten sich gegenseitig, bearbeiteten die gestellten Aufgaben meist gemeinsam und wechselten sich oft an der Tastatur ab. Innerhalb einer anderen Klasse kann die hier sehr beliebte Partnerarbeit jedoch dazu führen, dass gut die Hälfte der Lernenden nicht aktiv am Unterrichtsgeschehen beteiligt ist.

Der Rechner kann Lernende zu einem Vorgehen nach Versuch und Irrtum verleiten. Einige Schüler der Klasse bekamen so die einzelnen Tests zum Laufen und erhielten richtige Ergebnisse, erfassten die mathematischen Hintergründe aber kaum. Individuelle Hilfen und Fragen brachten Schüler dazu, die Prozeduren hinterfragend zu erzeugen. Ein anderer Ausweg wäre das schriftliche Erstellen einer Prozedur. Dies wird zu einer eingehenden Beschäftigung mit den Voraussetzungen für einzelne Testentscheidungen führen.

Die durch den Rechnereinsatz gewonnenen Ergebnisse müssen von den Lernenden festgehalten werden. Viele beschäftigten sich anfangs mit dem Abschreiben einzelner Prozeduren und verloren so wertvolle Unterrichtszeit. Die erstellten Programme sollten den Schülern auf Diskette und auch in schriftlicher Form zur Verfügung gestellt werden. Bewährt haben sich außerdem Arbeitsblätter mit Fragen, die Freiräume für Antworten enthalten. Eine häusliche Nacharbeit wird dadurch begünstigt.<sup>125</sup>

Beachten sollte man, dass die Schüler die Notwendigkeit des CAS – Einsatzes erkennen und die Ergebnisse auf das zu erreichende Unterrichtsziel beziehen. Am Anfang hatte es den Anschein, als ob einige diese Stunden als völlig losgelöst vom normalen Unterricht betrachteten. Als hilfreich erwiesen sich Vororientierungen und die ausführliche Diskussion der Ergebnisse.

Der Computereinsatz im Unterricht ist kein Allheilmittel. Bei den meisten<sup>126</sup> trug die Verwendung eines CAS zu einer guten Arbeitseinstellung innerhalb der vom Rechner geprägten Stunden bei, jedoch übertrug sich das Engagement nicht auf die eher klassischen Unterrichtsstunden und auf die häusliche Lernbereitschaft.

<sup>123</sup> Ein Schüler programmierte das *Sieb des Eratosthenes* in MuPAD, zwei andere beschäftigten sich mit dem Programm *prime95* zur Überprüfung von *Mersenneschen Primzahlen* (Das Programm befindet sich auf der CD.) u.s.w.

<sup>124</sup> Viele Schüler testeten immer größere Zahlen. Die Rechenzeiten nahmen unakzeptable Ausmaße an.

<sup>125</sup> Dies war z.B. bei einigen leistungsschwächeren Schülern erkennbar, die sich fleißig auf die Kontrolle vorbereitet hatten.

<sup>126</sup> Es gab einen Schüler, dem der Rechnereinsatz überhaupt nicht gefiel.

### **3. 6. Ausbau des Themas**

Der Einblick in das spezielle Gebiet der Primzahltests erscheint im Rahmen der Schulmathematik ausreichend. Die wesentliche Eigenschaft aktueller Tests, Wahrscheinlichkeitsaussagen zu treffen, und die Legitimation solcher Tests können innerhalb dieser Unterrichtseinheit genügend erörtert werden. Ansatzpunkte für die Ausweitung der Primzahlthematik ergeben sich jedoch viele. So könnten z.B. verschiedene Aspekte der Primzahlen<sup>127</sup>, das Generieren großer Primzahlen, die Faktorisierung großer Zahlen und Chiffrierverfahren<sup>128</sup> Einzug in den Mathematikunterricht halten. Alle diese Teilgebiete bieten die Möglichkeit des Einsatzes moderner Rechentechnik, geben ausreichend Gelegenheiten, das selbständige Problemlösen zu fördern, sind eng mit der aktuellen Forschung verbunden und dienen dem exemplarischen Erfahren der Bedeutung der Mathematik.

---

<sup>127</sup> Themenkomplexe findet man z.B. bei Fegert 1993, 16.

<sup>128</sup> Vgl. Schulz 1993, 56ff.

## 4. Zusammenfassung

Die Unterrichtseinheit «Primzahltests» wurde in einer 11. Klasse (Förderklasse Mathematik) am Ende des Schuljahres erprobt. Die fachlichen Grundlagen wurden weitestgehend in klassischen Unterrichtsformen und die Primzahltests mit Unterstützung eines Computeralgebra-Systems erarbeitet.

Der Unterrichtsversuch ergab, dass der gewählte Inhalt nur bedingt für den Unterricht einer 11. Klasse des Gymnasiums geeignet ist. Neben dem Bestehen wichtiger Rahmenbedingungen (Möglichkeit der Nutzung eines Computerkabinetts, großzügiger Zeitrahmen u.s.w.) sollte die Klasse aus leistungsstarken Schülern zusammengesetzt sein. Unter diesen Voraussetzungen wird der vorgestellte Unterrichtsgang einen erheblichen Beitrag zur Vermittlung von grundlegenden Denk- und Arbeitsweisen der Mathematik einbringen und die Erfahrungswelt der Lernenden bereichern.

Die behandelten Primzahltests verlangen den sinnvollen Einsatz eines CAS. Einerseits sind Beispielrechnungen nur mit einem solchen System möglich, andererseits können die Testbedingungen und Eigenschaften der Primzahltests gut in der praktischen Tätigkeit, dem Umsetzen der Tests in einem CAS, erarbeitet werden.

Die Verwendung eines Computeralgebra-Systems im Unterricht, vor allem der Ersteinsatz, bringt einige Probleme mit sich, die Beachtung finden müssen.

Ein Computeralgebra-System ist geeignet, viele Schüler für die Auseinandersetzung mit mathematischen Sachverhalten zu begeistern. Das Erwecken von Interesse am Fach muss eine essentielle Aufgabe des Mathematikunterrichts sein.

Die Zahlentheorie mit ihren vielen einfachen Fragestellungen, die das logische Denken energisch fordern, sollte einen größeren Raum innerhalb der Schulmathematik einnehmen.

Gauß formulierte:

„Die Höhere Arithmetik bietet einen unerschöpflichen Reichthum an interessanten Wahrheiten dar, und zwar an solchen, die nicht vereinzelt, sondern in innigem Zusammenhange stehen, und immer neue, ja unerwartete Verknüpfungen erkennen lassen, je weiter die Wissenschaft sich ausbildet“<sup>129</sup>.

---

<sup>129</sup> Remmert 1995, 7.

## 5. Literaturverzeichnis

### 5. 1. Bücher, Zeitschriften und Bildungspläne

**Bachmair**, Gerd: Unterrichtsanalyse: Verfahren und Fragestellungen zur Planung, Durchführung und Auswertung von Unterrichtsbeobachtungen. Weinheim: Beltz, **1974**, 11-78.

**Bartholomé**, Andreas; Rung, Josef; Kern, Hans: Zahlentheorie für Einsteiger. Braunschweig: Vieweg, **1995**.

**Bundschuh**, Peter: Einführung in die Zahlentheorie. 4. Auflage. Berlin: Springer, **1998**.

**Christmann**, Norbert: Einführung in die Mathematik-Didaktik. Paderborn: Schöningh, **1980**, 15-100.

**Conway**, John H.; Guy, Richard K.: Zahlenzauber: Von natürlichen, imaginären und anderen Zahlen. Basel: Birkhäuser, **1997**, 143-168.

**Dobermann**, Heike: Hausarbeit im Rahmen der ersten Staatsprüfung für das Lehramt am Gymnasium: Primzahltests mittels Lucas-Folgen. **1999**.

**Fegert**, Karl; Bernhard, Leeb; Brandt, Volker: Erfahrungen mit dem Kurs „Primzahlen“ der Schülerakademie. In: Mathematik lehren: Primzahlen I. Friedrich Verlag, Nr. 57, **1993**, 14-16.

**Fuchssteiner**, Benno; Wiwianka, Waldemar: MuPAD: Benutzerhandbuch. Basel: Birkhäuser, **1993**.

**Glatfeld**, Martin: Zur Einführung in das Heft „Primzahlen I“. In: Mathematik lehren: Primzahlen I. Friedrich Verlag, Nr. 57, **1993a**, 4.

**Glatfeld**, Martin: Konzeptionelle Bemerkungen zur unterrichtlichen Behandlung von Euklids Beweis der Unendlichkeit der Primzahlmenge. In: Mathematik lehren: Primzahlen I. Friedrich Verlag, Nr. 57, **1993b**, 5-7.

**Heymann**, Hans Werner: Allgemeinbildung und Mathematik. Weinheim: Beltz, **1996**, 131-280.

Ministerium für Bildung, Wissenschaft und Kultur Mecklenburg-Vorpommern (Hrsg.): **Rahmenplan** für das Gymnasium: **Sekundarstufe I**. Mathematik. **1997**.

Ministerium für Bildung, Wissenschaft und Kultur Mecklenburg-Vorpommern (Hrsg.): **Rahmenplan** für das Gymnasium: **Sekundarstufe II**. Mathematik. **1999**.

**Oevel**, Walter; Postel, Frank; Rüscher, Gerald; Wehmeier, Stefan: Das MuPAD Tutorium. Berlin: Springer, **2000**.

**Padberg**, Friedhelm: Elementare Zahlentheorie. 2. Auflage. Heidelberg: Spektrum, **1999a**.

**Padberg**, Friedhelm: Zahlentheorie und Arithmetik. Heidelberg: Spektrum, **1999b**.

**Remmert**, Reinhold; Ullrich, Peter: Elementare Zahlentheorie. 2. Auflage. Basel: Birkhäuser, **1995**, 1-235.

**Ribenboim**, Paulo: The new book of prime number records. 3. Auflage. New York: Springer, **1996**, 1-178.

**Riesel**, Hans: Prime numbers and computer methods for factorization. 2. Auflage. Boston: Birkhäuser, **1994**, 1-140.

**Scheu**, Günter: Entdeckungen in der Menge der Primzahlen mit DERIVE. In: Praxis der Mathematik. Aulis Verlag, Heft 3/34, **1992**, 119-122.

**Schulz**, Ralph-Hardo: Primzahlen in öffentlichen Chiffrierverfahren. In: Mathematik lehren: Primzahlen II. Friedrich Verlag, Nr. 61, **1993**, 56-64.

**Winning**, Anita: Ein Weg zu den Primzahlen. In: Mathematik lehren: Primzahlen II. Friedrich Verlag, Nr. 61, **1993**, 18-20.

**Winter**, Heinrich: Entdeckendes Lernen im Mathematikunterricht: Einblicke in die Ideengeschichte und ihre Bedeutung für die Pädagogik. 2. Auflage. Braunschweig: Vieweg, **1991**, 22-32.

**Zech**, Friedrich: Grundkurs Mathematikdidaktik: Theoretische und praktische Anleitungen für das Lehren und Lernen von Mathematik. 9. Auflage. Weinheim: Beltz, **1998**.

## 5. 2. Quellen aus dem Internet

Die angegebenen Quellen schließen weiterführende Links ein.

<http://www.primzahlen.de/> (deutsche Primzahlseite).

[http://www.learnetix.de/learnetix/mathe/primzahlen/primzahlen\\_einstieg.html](http://www.learnetix.de/learnetix/mathe/primzahlen/primzahlen_einstieg.html) (deutsche Primzahlseite).

<http://www.utm.edu/research/primres/> (Primzahlen und Primzahltests).

<http://www.mersenne.org/prime.htm> (Mersenne-Homepage; Primzahlsuchmaschine).

<http://www.isthe.com/chongo/tech/math/prime/> (*Mersennesche Primzahlen*).

<http://www.mathematik.uni-marburg.de/~jentschk/prime.html> (*Mersennesche Primzahlen*).

<http://www.jeckle.de/mersenne.html> (*Mersennesche*, Titanische und Gigantische Primzahlen).

<http://www.gomeck.de/welt-der-zahlen.html> (Wissenswertes über Zahlen).

<http://www.kinds-of-numbers.de/> (Zahlenarten).

<http://www.theory-of-numbers.de/> (Beweise zur *Unendlichkeit der Primzahlmenge*).

<http://die.antimaterie.de/lochfrass/inhalt.html> (Kongruenzrechnung; Primzahltests; Faktorisierungsverfahren).

<http://www.devalco.de/> (*Kleiner Satz von Fermat*).

[http://www.wiwi.uni-bielefeld.de/StatCompSci/lehre/material\\_spezifisch/statalg00/historisch/histnet.html](http://www.wiwi.uni-bielefeld.de/StatCompSci/lehre/material_spezifisch/statalg00/historisch/histnet.html) (*Euklidischer Algorithmus*; *Sieb des Eratosthenes*; Primzahltests).

<http://www.muenchenbach.de> (Von Primzahlen zur Kryptographie).

[http://www.scruznet.com/~luke/lit/lit\\_068s.htm](http://www.scruznet.com/~luke/lit/lit_068s.htm) (Lucas über *Fermatsche* und *Mersennesche Primzahlen*).

[http://www.math.umn.edu/~garrett/js/list\\_pr.html](http://www.math.umn.edu/~garrett/js/list_pr.html) (Java Applet: Primzahlgenerator).

<http://www.faust.fr.bw.schule.de/mhb/eratosib.htm> (Java Applet: *Sieb des Eratosthenes*).

[http://www.cims.nyu.edu/~iserov/java/prime\\_test.html](http://www.cims.nyu.edu/~iserov/java/prime_test.html) (Java Applet: Primfaktorzerlegung).

<http://members.aol.com/m3021377/private/latimes.html> (Student Finds Largest Prime Number Ever).

<http://www.mathematik.ch/mathematiker.html> (Kurzbiographien bedeutender Mathematiker).

<http://www.mathe.tu-freiberg.de/~hebisch/cafe/lebensdaten.html> (Lebensdaten bedeutender Mathematiker).

<http://www.mathe.tu-freiberg.de/~hebisch/cafe/zitate.html> (Zitate zur Mathematik).

<http://www.pirabel.de/zitate.htm> (Zitate zur Mathematik).

<http://www.sodis.de/> (Software-Dokumentations- und Informations-System).

<http://www.sciface.com/> (Vertrieb von MuPAD; Anleitungen zu MuPAD).

<http://www.mupad.de/> (Vertrieb von MuPAD; Anleitungen zu MuPAD).

[http://www.art1.it-netservice.de/privat/mupad/MuPAD-dt\\_Doku.html](http://www.art1.it-netservice.de/privat/mupad/MuPAD-dt_Doku.html) (Anleitung zu MuPAD).

**5. 3. Titelbild**

Im Jahr 1963 wies Prof. Donald B. Gillies<sup>130</sup> mit dem Supercomputer ILLIAC-II der Universität Illinois (U.S.A.) nach, dass die Zahlen 9689, 9941 und 11213 in der Mersenneschen Formel Primzahlen liefern. Das Postamt in Urbana Illinois ehrte die Entdeckung der 23. Mersenneschen Primzahl mit dem dargestellten Poststempel.

**Bildquelle:**

Remmert, Reinhold; Ullrich, Peter: Elementare Zahlentheorie. 2. Auflage. Basel: Birkhäuser, 1995, 40.

---

<sup>130</sup> Prof. Donald B. Gillies (1928 – 1975)

## 6. Symbole und Bezeichnungen

CAS	Computeralgebra-System
MuPAD	<b>M</b> ulti <b>P</b> rocessing <b>A</b> lgebra <b>D</b> ata <b>T</b> ool
PC	<b>P</b> ersonalcomputer
RSA	Verschlüsselungsverfahren von <b>R</b> ivest, <b>S</b> hamir und <b>A</b> dleman
EA	Einzelarbeit <sup>131</sup>
UG	Unterrichtsgespräch
LV	Lehrervortrag
HA	Hausaufgabe
PA	Partnerarbeit
$Z$	Menge der ganzen Zahlen
$N$	Menge der natürlichen Zahlen
$N_0$	Menge der natürlichen Zahlen zuzüglich 0
$\text{ggT}(a, b)$	größter gemeinsamer Teiler von $a$ und $b$
$\varphi$	Eulersche $\varphi$ -Funktion
$a \mid b$	$a$ teilt $b$
$a \equiv b \pmod{m}$	$a$ kongruent $b$ modulo $m$
$a \not\equiv b \pmod{m}$	$a$ inkongruent $b$ modulo $m$
$\bar{a}$	Restklasse modulo $m$
$[x]$	größte ganze Zahl kleiner gleich $x$

---

<sup>131</sup> Innerhalb des Unterrichts sind bei der Einzelarbeit Hilfen von Mitschülern meist gestattet. Strenge selbständige Schülerarbeit wird insbesondere dann gefordert, wenn die erbrachten Leistungen benotet werden sollen.

## 7. Anhang

### 7. 1. Inhalt der beiliegenden CD-ROM

Die meisten Dokumente liegen im pdf-Format vor, so dass der Acrobat Reader benötigt wird. Einige Dateien sind mit WinZip gepackt und müssen vor Benutzung extrahiert werden.

Beide Programme sind kostenlos im Internet erhältlich:

- Acrobat Reader <http://www.adobe.com>
- WinZip <http://www.winzip.com>

- Im Verzeichnis «Arbeitsblaetter» befinden sich alle Arbeitsblätter und Folien, die während des Unterrichtsversuches zum Einsatz kamen. Einige sind sowohl in Folienform als auch in kompakter Form für die Hand des Schülers formatiert. Außerdem enthalten viele Dateien Lösungen zu den gestellten Aufgaben.
- Im Verzeichnis «Hausarbeit» befindet sich diese Arbeit.
- Im Verzeichnis «MuPAD\_Anleitungen» befinden sich Kurzanleitungen und Beispieldateien zu MuPAD. Außerdem ist eine komplette Dokumentation zu MuPAD 2.0 enthalten.
- Im Verzeichnis «MuPAD\_Installation» befinden sich die Installationsdateien zu den Versionen MuPAD 1.4.2 und MuPAD 2.0.<sup>A1</sup> Es sind jeweils die kostenfreie Light-Version und die kostenpflichtige Pro-Version<sup>A2</sup> vorhanden. Beide müssen bei SciFace registriert werden, wobei die letztere 30 Tage lang getestet werden kann.
- Im Verzeichnis «MuPAD\_Notebooks» befinden sich die Lösungen der Arbeitsblätter im mnb-Format<sup>A3</sup> und im txt-Format.
- Im Verzeichnis «Primzahlen» befinden sich eine Primzahlliste bis 600011 und eine Übersicht über die größten bekannten Primzahlen mit Angaben zur Entdeckung.
- Im Verzeichnis «Programme» befinden sich der Primzahltester der Mersenne-Homepage, ein Programm zum Erstellen von Primzahllisten und ein Tool, das Primzahlen zum Musizieren nutzt.

---

<sup>A1</sup> Die Schüler nutzten MuPAD Light 1.4.2 zu Hause. Im Computerraum der Schule war die deutsche Version von MuPAD Pro 1.4.2 installiert. MuPAD 2.0 ist erst seit kurzem erhältlich.

<sup>A2</sup> Die Pro-Version ist in deutscher und englischer Sprache verfügbar.

<sup>A3</sup> MuPAD-Notebook-Format

## **7. 2. Arbeitsblätter, Folien und Ergänzungen**

Die Arbeitsblätter und Folien sind entsprechend der Reihenfolge ihres Einsatzes im Unterricht sortiert.

## Primzahltests in aktuellen Computeralgebra-Systemen

In den Hilfen der Computeralgebra-Systeme MuPAD, Maple, Mathematica und DERIVE findet man zu den dort verwendeten Primzahltests folgende Beschreibungen:

„isprime ist ein stochastischer Primzahltest. Die Funktion gibt TRUE zurück, wenn n eine Primzahl oder eine starke Pseudoprimzahl für zehn zufällig gewählte Basen ist, FALSE sonst.“<sup>A4</sup>

„The function isprime is a probabilistic primality testing routine. It returns false if n is shown to be composite within one strong pseudo-primality test and one Lucas test and returns true otherwise. If isprime returns true, n is very probably prime (...) No counter example is known and it has been conjectured that such a counter example must be hundreds of digits long.“<sup>A5</sup>

„In Mathematica Version 2.0, PrimeQ[n] uses the Rabin strong pseudoprime test and the Lucas test. This procedure has been proved correct for all  $n < 2,5 \cdot 10^{10}$ . As of 1990, however, the procedure has not been proved correct for larger n, and it is conceivable that it could claim that a composite number was prime (though not vice-versa). Nevertheless, as of 1990, no example of such behavior is known.“<sup>A6</sup>

„PRIME?(n) wird zu true vereinfacht, falls n eine Primzahl ist und zu false, falls n keine Primzahl ist (...) Die zum Herausfinden der Primzahlen verwendeten Methoden beinhalten das Primzahlensieb, den Rabin-Miller Primzahlen-Test und den Lucas Primzahlen-Test.“<sup>A7</sup>

---

<sup>A4</sup> MuPAD PRO Version 1.4.2.

<sup>A5</sup> Maple V Release 4 Version 4.00c.

<sup>A6</sup> Mathematica for Windows Version 2.2.

<sup>A7</sup> DERIVE für Windows Version 5.02.

MuPAD (**M**ulti **P**rocessing **A**lgebra **D**ata **T**ool) ist ein Computeralgebra-System, das vom Institut AUTOMATH (Institut für **A**utomatisierung und Instrumentelle **M**athematik) an der Universität Paderborn entwickelt wurde. Mit einem CAS (Computeralgebra-System) können symbolische Berechnungen (Berechnungen mit mathematischen Objekten) durchgeführt werden. Solche Objekte können Zahlen, Polynome, Gleichungen, Formeln oder beliebige andere mathematische Objekte sein. Symbolische Berechnungen mit Zahlen werden im Gegensatz zu numerischen exakt durchgeführt. Schon dieser Umstand macht ein CAS für die Mathematik so wertvoll.

Man kann ein CAS wie MuPAD interaktiv bedienen, d.h. man gibt eine Anweisung (z.B. die Multiplikation zweier Zahlen) ein und wartet, bis MuPAD das Ergebnis berechnet hat und auf dem Bildschirm ausgibt. Des Weiteren bietet MuPAD ein Konzept zum objektorientierten Programmieren. Man hat u.a. also die Möglichkeit, eigene Funktionen und Prozeduren zu definieren.

Wir werden uns am Anfang mit der interaktiven Bedienung, also mit den Funktionen befassen, die von MuPAD zur Verfügung gestellt werden. Erste Hinweise zum komfortableren Umgang mit MuPAD Light finden Sie auf der Rückseite.

Scheuen Sie sich nicht, mir Fragen zu stellen, falls Sie an irgendeiner Stelle nicht zurecht kommen sollten. Wenn Sie mir evtl. Schwierigkeiten nicht mitteilen, werden sich die Probleme nur unnötig anhäufen.

## Das Rechnen mit Zahlen

Starten Sie MuPAD Light. Nach kurzer Zeit erscheint ein Prompt (kleiner roter Punkt). MuPAD wartet auf Ihre Eingabe. Die Eingaben werden im folgenden jeweils mit einem kleinen schwarzen Punkt kennzeichnen. Lesen Sie bitte genau und achten Sie darauf, die Eingaben exakt zu übernehmen. Falls Sie sich doch einmal vertippen sollten, geben Sie die Eingabe einfach beim nächsten Prompt ein.

Tippen Sie folgendes ein und bestätigen Sie mit **ENTER**:

- **1 + 5/2;** Achten Sie auf das **Semikolon**, das fast jede Eingabe beendet.
- **(1+(5/2\*3))/(1/7+7/9)^2;** Das Symbol **^** steht für das Potenzieren.
- **1234^123;** liefert die 123-te Potenz von 1234. Eingabe: **1234^Leertaste123;**

Beachten Sie, dass das Zeichen **\** (Backslash) bedeutet, dass das Ergebnis in der nächsten Zeile fortgesetzt wird.

- **100!;** liefert die Fakultät von 100 ( $100! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot 100$ )

Welche größte Fakultät kann Ihr Taschenrechner berechnen? Wo liegen die Grenzen von MuPAD?

(Sie werden feststellen, dass MuPAD hier keine Grenzen kennt. Die Limits werden allein durch den zur Verfügung stehenden Hauptspeicher Ihres Rechners gesetzt. Die Berechnungen dauern dementsprechend bei großen Zahlen länger.)

- **isprime(123456789);** prüft, ob die Zahl 123456789 eine Primzahl ist.
- **ifactor(123456789);** liefert die Primfaktoren von 123456789.

Als Ergebnis gibt MuPAD die Liste [1, 3, 2, 3607, 1, 3803, 1]. Es gilt:  $123456789 = 1 \cdot 3^2 \cdot 3607^1 \cdot 3803^1$ .

- **sqrt(56);** sqrt...Wurzel

MuPAD gibt  $2 \cdot 14^{\frac{1}{2}}$  aus, rechnet also immer noch exakt ( $\sqrt{56}$  ist eine irrationale Zahl.) und zieht zur Vereinfachung nur partiell die Wurzel.

Will man einen numerischen Näherungswert von  $\sqrt{56}$  wissen, so nutze man den Befehl float:

- **float(sqrt(56));**

MuPAD liefert 7.483314773; berechnet  $\sqrt{56}$  also auf 9 Dezimalen genau.

Das kann ein Taschenrechner auch! Erhöhen wir die Stellenzahl!

- **DIGITS:=100; float(sqrt(56));** Beachten Sie die **GROßSCHREIBUNG** von DIGITS!

Der Standardwert von DIGITS für numerische Berechnungen ist 10. Beachten Sie, dass Sie den Wert von DIGITS selbst wieder zurückzusetzen müssen. Spätere Berechnungen könnten sonst sehr lange dauern.

- **DIGITS:=10;**

Falls Sie nur für eine Berechnung mehrere Dezimalstellen benötigen, sollten Sie sich angewöhnen, DIGITS im gleichen Atemzug mit der Ausgabe wieder auf den Standardwert zu setzen:

- **DIGITS:=100: float(sqrt(56)); DIGITS:=10:** Beachten Sie **Doppelpunkt** und **Semikolon**.

Die Ergebnisse der Eingaben, die mit einem Doppelpunkt abgeschlossen werden, berechnet MuPAD, aber gibt sie nicht auf den Bildschirm aus.

Die Funktionen von MuPAD liefern numerische Ergebnisse, wenn ihr Argument eine Gleitkommazahl ist:

- **sqrt(56.0); sqrt(56);** Beachten Sie, dass das Komma von 56,0 als **Punkt** geschrieben wird.
- **sin(3.14), cos(3.14), tan(3.14);** liefert **0.001592652916, -0.999987317, -0.001592654936**
- **sin(PI), cos(PI), tan(PI);** liefert **0, -1, 0** **GROßSCHREIBUNG** von PI.

Mit Hilfe der Konstanten  $\pi$  kann MuPAD exakt rechnen.

Lassen Sie sich einige Stellen der Eulerschen Zahl  $e$  anzeigen und berechnen Sie den natürlichen Logarithmus von  $e$  numerisch und exakt:

- **DIGITS:=200: float(E); float(ln(E)); ln(E); DIGITS:=10;** **GROßSCHREIBUNG** von E

### Aufgabe:

Berechnen Sie  $\sqrt{27} - 4\sqrt{3}$  und  $\cos\left(\frac{\pi}{8}\right)$  exakt.

Ermitteln Sie auf 22 Stellen nach dem Komma genaue Näherungen.

### Ergebnisse

$\sqrt{27} - 4\sqrt{3} =$  \_\_\_\_\_ Die 20., 21. und 22. Stelle von  $\sqrt{27} - 4\sqrt{3}$  lautet: \_\_\_\_\_.

$\cos\left(\frac{\pi}{8}\right) =$  \_\_\_\_\_ Die 20., 21. und 22. Stelle von  $\cos\left(\frac{\pi}{8}\right)$  lautet: \_\_\_\_\_.

### Besonderheiten von MuPAD Light

In MuPAD Light sind leider nicht alle Funktionen, die in MuPAD Pro implementiert sind, vorhanden. Die Bedienung von MuPAD Light ist außerdem etwas umständlicher.

Sie werden sicherlich schon bemerkt haben, dass man in MuPAD Light eine einmal bestätigte Eingabe nicht mehr editieren (verändern) kann. Dies ist besonders ärgerlich, wenn man Fehler bei der Eingabe macht, da man dann alles völlig neu schreiben müsste.

Deshalb ist es sehr hilfreich, sich der Zwischenablage von Windows zu bedienen.

Kopieren Sie die Eingabe **DIGITS:=200: float(E); float(ln(E)); ln(E); DIGITS:=10;** und fügen Sie die Eingabe beim letzten Prompt wieder ein:

1. Markieren von **DIGITS:=200: float(E); float(ln(E)); ln(E); DIGITS:=10;**
  - Ziehen Sie bei gedrückter linker Maustaste den Mauspfel über den zu markierenden Text *oder*
  - Bewegen Sie den Mauspfel auf den Prompt der Zeile und klicken Sie einmal links *oder*
  - Bewegen Sie den Cursor an den Anfang der Zeile und drücken Sie **SHIFT + ENDE** (**SHIFT + ENDE** bedeutet: Auf der Tastatur **SHIFT**-Taste gedrückt halten und **ENDE** drücken) *oder*...
2. Drücken Sie **STRG + C**
3. Bewegen Sie den Cursor an den letzten Prompt und drücken Sie **STRG + V**.
4. Verändern Sie die Eingabe so, dass Ihnen jetzt nur noch 100 Dezimalstellen angezeigt werden.

Je besser Sie das Kopieren von Textteilen beherrschen, desto leichter wird Ihnen die Arbeit in vielen Programmen fallen.

Speichern Sie nun zum Abschluss Ihre MuPAD-Sitzung ab. Beim ersten Mal sollten Sie einen neuen Ordner erstellen (Punkte 2 und 3), in den Sie Ihre MuPAD-Sitzungen abspeichern können.

1. Wählen Sie im Menü **File** den Unterpunkt **Save As**.
2. Klicken Sie mit der Maus auf das Symbol „**Neuen Ordner erstellen**“.
3. Geben Sie einen Namen für Ihren neuen Ordner ein: **MuPAD**
4. Wechseln Sie durch Doppelklick auf den Ordernamen in den Ordner **MuPAD**.
5. Geben Sie bei Dateiname den Namen für die MuPAD-Sitzung ein: **Zahlen**
6. Klicken Sie auf **Speichern**

Die entworfenen Dateien können in MuPAD Light nur als Textdateien abgespeichert und nicht wieder geladen werden. Man kann aber auch hier die Kopierfunktionen der Zwischenablage in Verbindung mit einem Editor nutzen, um ältere Dateien zu bearbeiten. Doch dazu später.

## Symbolisches Rechnen

Ein symbolischer Ausdruck in MuPAD kann unbestimmte Größen (Bezeichner oder Variablen) enthalten, mit denen gerechnet werden kann.

### Lösen von Gleichungen

- `solve(2*x + y = 3, x);` löst die Gleichung  $2x + y = 3$  nach  $x$  auf
- `solve(2*x + y = 3, y);` löst die Gleichung  $2x + y = 3$  nach  $y$  auf

Da es umständlich ist, größere Gleichungen jeweils neu einzugeben (Nutzen Sie hier das Kopieren!), kann man die Gleichung auch erst definieren:

- `Gleichung := 2 * x + y = 3;` Dem Bezeichner „Gleichung“ wird  $2x + y = 3$  zugewiesen.
- `Gleichung;` liefert die Gleichung  $2x + y = 3$
- `solve(Gleichung, x), solve(Gleichung, y);` liefert die schon bekannten Auflösungen

Enthält eine Gleichung nur eine Unbekannte, so kann auf die Angabe der Variablen, nach der die Gleichung aufzulösen ist, verzichtet werden:

- `solve(2 * x^2 + 3 * x - 25 = 0);` liefert die Nullstellen von  $y = 2x^2 + 3x - 25$

Dagegen muss man bei mehreren Unbekannten die Variable, nach der aufgelöst werden soll, angeben:

- `solve(x^2 + p*x + q = 0, x);` liefert die Nullstellen von  $y = x^2 + px + q$

Will man die numerischen Näherungswerte wissen, so nutze man den schon bekannten Befehl `float`:

- `float(solve(2 * x^2 + 3 * x - 25 = 0));` Nullstellen von  $y = 2x^2 + 3x - 25$  numerisch

MuPAD stören auch komplexe Lösungen nicht:

- `solve(x^2 - 2 * x + 2 = 0);`

Wie sieht es mit Polynomen höheren Grades aus?

- `solve(x^3 - 8 * x^2 + x + 42 = 0);`
- `solve(x^4 + 12 * x^3 - 159 * x^2 + 62 * x + 840 = 0);`
- `solve(x^5 - 41 * x^4 - 795 * x^3 - 8489 * x^2 - 2446 * x - 44520 = 0);`

Für Polynome 3. Grades gibt es noch komplizierte Lösungsformeln für die Berechnung der Nullstellen. Ab 4. Grades muss die Mathematik im allgemeinen Fall passen. Trotzdem findet MuPAD für viele Gleichungen noch exakte Lösungen. Die Gleichung 5. Grades kann MuPAD nicht mehr exakt lösen. Es gibt jedoch Näherungsverfahren, die weiterhelfen.

### Lösen von Gleichungssystemen

- `Gleichungen1:={x + y = a, x - a * y = b};` Dem Bezeichner „*Gleichungen1*“ werden  $x + y = a$  und  $x - ay = b$  zugewiesen.
- `solve(Gleichungen1, {x, y});` löst das Gleichungssystem nach  $x$  und  $y$  auf

Benötigt man die Gleichungen später nicht mehr, genügt es, die Gleichungen direkt bei `solve` anzugeben:

- `solve({x + y = 2, x - 2 * y = 3}, {x, y});` liefert  $x = \frac{7}{3}$  und  $y = -\frac{1}{3}$  als Lösungen

- `Gleichungen2 := {x^2 + y = 1, x - y = 2}; solve(Gleichungen2, {x, y});`

Liefert die beiden Lösungen  $\left( y = x - 2; x = \frac{1}{2}\sqrt{13} - \frac{1}{2} \right)$  und  $\left( y = x - 2; x = -\frac{1}{2}\sqrt{13} - \frac{1}{2} \right)$ .

Das letzte Beispiel befriedigt natürlich noch nicht, da  $y$  jeweils noch nicht vollständig berechnet wurde, also noch von der Variablen  $x$  abhängig ist. Deshalb übergibt man die Option `BackSubstitution = TRUE`, damit MuPAD die Lösungen ineinander einsetzt:

- `Loesungen := solve(Gleichungen2, {x, y}, BackSubstitution = TRUE);` Beachte *Loesungen*!

Nun wird die Lösung für  $x$  auch sofort in  $y$  eingesetzt und MuPAD liefert  $\left( y = \frac{1}{2}\sqrt{13} - \frac{5}{2}; x = \frac{1}{2}\sqrt{13} - \frac{1}{2} \right)$

und  $\left( y = -\frac{1}{2}\sqrt{13} - \frac{5}{2}; x = -\frac{1}{2}\sqrt{13} - \frac{1}{2} \right)$ .

Überprüfen wir die Lösungen! Dazu müssen wir die Ergebnisse in die Gleichungen  $x^2 + y = 1$  und  $x - y = 2$  einsetzen. Fangen wir mit der ersten Lösung  $\left(y = \frac{1}{2}\sqrt{13} - \frac{5}{2}; x = \frac{1}{2}\sqrt{13} - \frac{1}{2}\right)$  an!

- **subs(Gleichungen2, op(Loesungen, 1));** (**op(Loesungen, 1)** greift auf die 1. Lösung zu.)

setzt in den Gleichungen2 die 1. Lösung ein und liefert  $2 = 2$  und  $\left(-\frac{1}{2}\sqrt{13} - \frac{1}{2}\right)^2 - \frac{1}{2}\sqrt{13} - \frac{5}{2} = 1$ .

Die erste Bestätigung ist zufriedenstellend.

Die Probe in der 2. Gleichung sollte MuPAD aber auch alleine auflösen.

- **simplify(subs(Gleichungen2, op(Loesungen, 1)));** (**simplify** vereinfacht Ausdrücke)

Wir erhalten  $1 = 1$  und  $2 = 2$ , was die Gültigkeit der 1. Lösung bestätigt.

Nun noch die Überprüfung der 2. Lösung:

- **simplify(subs(Gleichungen2, op(Loesungen, 2)));**

Will man die numerischen Näherungslösungen wissen, so nutze man wieder float. Die Operation float allein würde jedoch nur die symbolische Lösung von **solve** weiterverarbeiten, was bei mehreren Lösungen MuPAD Schwierigkeiten bereitet. Mit der Operation **hold** wird die symbolische Rechnung unterdrückt und ausschließlich numerisch gerechnet:

- **float(hold(solve)(Gleichungen2, {x, y}));**

liefert  $(y = -4,3\dots; x = -2,3\dots)$  und  $(y = -0,697\dots; x = 1,302\dots)$

### Aufgaben:

1. Berechnen Sie die Nullstellen der Gleichung  $y = 5x^2 + 4x - 32$  exakt und numerisch.
2. Lösen Sie das folgende Gleichungssystem allgemein:

$$a + b + c + d + e = 1$$

$$a + 2b + 3c + 4d + 5e = 2$$

$$a - 2b - 3c - 4d - 5e = 2$$

$$a - b - c - d - e = 3$$

(Die Lösung enthält freie Parameter. Wie viele?)

### Entscheiden sie selbst, wie viel Übung Sie benötigen.

3. Lösen Sie S. 63/A151 a, b, d exakt und geben Sie für b) eine Näherungslösung an.
4. Lösen Sie S. 63/A152 exakt und ermitteln Sie Näherungslösungen (Variieren Sie auch die Stellenzahl der Genauigkeit.).
5. Lösen Sie S. 64/A159 exakt und ermitteln Sie Näherungslösungen.

Speichern Sie die Arbeit unter **Gleichungen.txt** ab.

## Funktionen und Manipulation von Ausdrücken

MuPAD unterscheidet zwischen „normalen“ algebraischen Ausdrücken (Funktionen von Argumenten) und Funktionen (Abbildungen: Argumente  $\rightarrow$  Wert). Es ist in manchen Fällen sinnvoller, Funktionen zu definieren.

### Definition von Funktionen

- **Funktion := x  $\rightarrow$  (x<sup>2</sup>);** definiert Funktion(x) = x<sup>2</sup> ( $\rightarrow$  ... „Minus“ und „größer als“)
- **Ausdruck := x<sup>2</sup>;** Hier wird keine Funktion definiert.

Es bestehen einige Unterschiede zwischen den beiden Definitionen. Der für uns zur Zeit wichtigste Vorteil der Funktionsdefinition gegenüber der Benutzung von Ausdrücken ist die Berechnung von Funktionswerten:

- **Funktion(8); Ausdruck(8);** Funktionswert von 8; Ausdruck(8) kann dies nicht.
- **f := x  $\rightarrow$  (cos(x)); g := x  $\rightarrow$  (x<sup>3</sup>);** definiert die Funktionen f(x) = cos(x) und g(x)=x<sup>3</sup>

Wir können nun Funktionen verketteten, d.h. ineinander verschachteln:

- **h := f @ g; h(x); h(5); float(h(5));** h(x) = f(g(x)) = cos(x<sup>3</sup>) (@ ... AltGr und Q drücken)

zeigt cos(x<sup>3</sup>) an, berechnet den Funktionswert h(5) exakt und numerisch.

Die Reihenfolge der Verkettung sollte beachtet werden!

- **h2 := g @ f; h2(x); h2(5); float(h2(5));** h2(x) = g(f(x)) = (cos(x))<sup>3</sup> = cos<sup>3</sup>(x).

Der Prozess der Verkettung kann beliebig oft fortgesetzt werden.

- **h3 := g@@3; h3(x); h3(5);** h3(x) = g(g(g(x)))= x<sup>27</sup>

Man kann Funktionen definieren, die von beliebig vielen Variablen abhängen:

- **f := (x, y)  $\rightarrow$  (x<sup>2</sup> + y<sup>2</sup>);**

Zu beachten ist hier, dass die oben definierte Funktion f(x) = cos(x) nun überschrieben wurde. Mit f(5); kann nun nicht mehr cos(5) berechnet werden. Die Funktion f erwartet jetzt 2 Parameter:

- **f(5); f(3,4); f(a,b+1);**

Zur Berechnung von Funktionswerten müssen Ausdrücke in Funktionen konvertiert werden. Will man die Funktion dann nicht wieder unter Eingabe des gesamten Ausdrucks definieren, kann man den Befehl *unapply* nutzen. Dieser Befehl muss vorher aus der Funktionsbibliothek *fp* geladen werden:

- **export(fp,unapply);** (nur einmal aufrufen, steht dann immer zur Verfügung)
- **Ausdruck := x<sup>3</sup> + sin(x);** definiert einen Ausdruck
- **Funktion := unapply(Ausdruck);** wandelt den Ausdruck in eine Funktion um
- **Ausdruck(5); Funktion(5);** Nur der zweite Befehl kann den Funktionswert berechnen.

### Manipulation von Ausdrücken

Eine sehr wichtige Aufgabe von einem CAS ist das Vereinfachen oder Umformen von Ausdrücken.

Der Befehl *expand* multipliziert Ausdrücke aus oder wendet die Additionstheoreme an:

- **expand(exp(x + y)); expand(sin(x - y)); expand(cos(x + y)); expand(tan(x + y));**

Statt x und y können natürlich wiederum Ausdrücke verwendet werden, z.B.:

- **expand(tan(x + 3\*PI/2));**

Mit *normal* werden rationale Ausdrücke zusammengefasst, also auf einen Nenner gebracht:

- **A1:= x/(1+x) - 2/(1-x); A2 := normal(A1);**

Gemeinsame Faktoren in Zähler und Nenner werden durch *normal* gekürzt:

- **A3 := x<sup>2</sup>/(x+y) - y<sup>2</sup>/(x+y); normal(A3);**

Umgekehrt wird ein rationaler Ausdruck durch *partfrac* in eine Summe rationaler Terme zerlegt:

- **partfrac(A2,x);** zerlegt den oben definierten Ausdruck A2

Der Befehl *simplify* ist ein universeller Vereinfacher. MuPAD versucht eine einfache Darstellung zu finden.

- **A4 := (exp(x) - 1)/(exp(x/2)+1); simplify(A4);**

Die Funktion *radsimp* vereinfacht Ausdrücke mit Wurzeln:

- **A5 := sqrt(4 + 2\*sqrt(3)); radsimp(A5);**

Der Befehl *Factor* überführt eine Summe in ein Produkt. (Schreiben Sie die Eingaben zum Vergleich auf!)

- **Factor(x<sup>3</sup>+3\*x<sup>2</sup>+3\*x+1); Factor(2\*x\*y-2\*x-2\*y+x<sup>2</sup>+y<sup>2</sup>); Factor(x<sup>2</sup>/(x+y)-z<sup>2</sup>/(x+y));**

### Aufgaben:

1. Berechnen Sie die f(22) und f(-13) exakt und numerisch.  $f(x) = x^3 - 2\sin^2(x) + 25$
2. Multiplizieren Sie die Ausdrücke  $(x + y)^2$ ,  $(x + y)^{100}$  und  $(x^2 - y)^5$  aus!
3. Überprüfen Sie die Additionstheoreme sin(x+y), cos(x-y) und tan(x-y).
4. Versuchen Sie, die Funktionen LB S. 225/C177 als Produkte darzustellen.
5. Bilden sie die Summe aus den Funktionen f<sub>1</sub>, f<sub>2</sub> und f<sub>3</sub> (LB S. 225/C177) und bringen Sie die Summe auf einen Nenner. (analog mit f<sub>4</sub>, f<sub>5</sub>, f<sub>6</sub> und f<sub>7</sub>, f<sub>8</sub>, f<sub>9</sub>)

Speichern Sie die Arbeit unter **Manipulation.txt** ab.

## Das Arbeiten mit Listen

MuPAD stellt Mengen, Folgen und Listen zum Zusammenfassen von Elementen zur Verfügung. Listen sind vor allem für das Programmieren ein gutes und notwendiges Hilfsmittel, wie später zu sehen sein wird.

Eine Liste ist eine geordnete Folge beliebiger MuPAD-Objekte, die in eckige Klammern eingeschlossen wird.

- **Liste := [a, 5, sin(x)^2 + 4, [a, b, c], hallo, 3/4, 3.9087];** (*[... AltGr + 8; ] ... AltGr + 9*)

In einer Liste können beliebige Objekte stehen. Wie [a, b, c] im Beispiel zeigt, kann eine Liste selbst wiederum Listen als Elemente enthalten. Listen können auch leer sein:

- **LeereListe := [ ];** definiert eine leere Liste

Der  $\$$ -Operators zur automatischen Erzeugung langer Folgen ist sehr hilfreich zur Konstruktion langer Listen:

- **Folge := n \$ n = 1..20; Liste1 := [Folge];**

Der erste Befehl definiert eine Folge natürlicher Zahlen, die dann zur Konstruktion der Liste herangezogen wird. Dies geht natürlich auch direkt, wie die folgenden drei Beispiele zeigen:

- **Liste1 := [n \$ n = 1..20]; Liste2 := [2^n \$ n = 1..20]; Liste3 := [x^i \$ i = 1..20];**

Die Anzahl der Elemente einer Liste kann mit der Funktion *nops* festgestellt werden, die Elemente können mit der Funktion *op* ausgelesen werden.

- **nops(Liste); nops(Liste1);** Anzahl der Elemente der ersten Liste: 7; liefert 20
- **op(Liste, 5); op(Liste1, 10);** liefert 5. Element der Liste *Liste*: hallo; liefert 10
- **op(Liste, 2..4); op(Liste);** 2., 3. und 4. Element der Liste *Liste*; alle Elemente der Liste *Liste*

Eine alternative Möglichkeit, einzelne Listenelemente zu erhalten, liefert der *Index*-Operator, der schneller auf die Listenelemente zugreift, als es die *op*-Funktion vermag. (Rechenzeit bei Verwendung von Schleifen!)

- **Liste[1]; Liste[6];** erstes Element der Liste *Liste* bzw. sechstes Element der Liste *Liste1*.

Man kann den Wert eines Elements einer Liste verändern:

- **Liste[1] := neu; Liste;**

Das erste Element der Liste *Liste* war vorher mit *a* belegt. Nun wurde diesem Element der Wert *neu* zugewiesen.

Das Entfernen eines Elements aus einer Liste erfolgt durch *unassign*:

- **unassign(Liste[1]); Liste; nops(Liste);** entfernt das erste Element  $\Rightarrow$  *Liste* enthält nur 6 Elemente

Mit der Funktion *append* können Elemente an eine Liste angehängt werden:

- **append(Liste, neu1);** hängt das Element *neu1* am Ende der Liste an, verändert sie nicht dauerhaft:
- **Liste; nops(Liste);** Die Liste enthält immer noch die alten 6 Elemente.

Man kann aber die durch *append* neu entstehende Liste einer anderen Liste oder derselben Liste zuweisen:

- **Liste3 := append(Liste, neu1);** neue Liste *Liste3* mit dem angehängten Element ist entstanden
- **Liste; nops(Liste); Liste3; nops(Liste3);**
- **Liste := append(Liste, neu1);** Die Liste *Liste* wurde nun dauerhaft verändert.
- **Liste; nops(Liste);** Die Liste *Liste* besteht nun wieder aus 7 Elementen.

Es ist möglich, gleichzeitig mehrere Elemente anzuhängen:

- **Liste := append(Liste, neu2, neu3, neu4); Liste; nops(Liste);**

Mit Hilfe des *Punkt*-Operators können zwei Listen zusammengefügt werden. (Reihenfolge der Listen beachten!)

- **Liste1.Liste3; Liste3.Liste1;**

Mit *sort* werden Listen sortiert. Numerische Werte werden ihrer Größe nach, Zeichenketten lexikographisch angeordnet:

- **L1 := [-1.23, 4, 3, 2, 1/2]; sort(L1);** sortiert die neu definierte Liste *L1*
- **L2 := ["A", "c", "abc", "a1", "aa", "C", "Abc"]; sort(L2);** sortiert die neu definierte Liste *L2*
- **L3 := [A, c, abc, a1, aa, C, Abc]; sort(L3);** sortiert die neu definierte Liste *L3*

Man beachte, dass die lexikographische Anordnung nur bei Benutzung von mit " erzeugten Zeichenketten verwendet wird (z.B. *L2*). Bei Namen von Bezeichnern wird u.a. die Länge der Namen berücksichtigt (siehe *L3*).

Die Funktion *select* dient dazu, Listenelemente mit bestimmten Eigenschaften aus einer Liste herauszufiltern. Die Funktion *select* liefert eine Liste mit allen Elementen, die die angegebene Eigenschaft erfüllen:

- **L4 := [n \$ n=1..20]; select(L4, isprime);** liefert alle Primzahlen der Liste *L4*

### Aufgaben:

1. Erstellen Sie eine Liste L1, welche die ersten 100 Quadratzahlen enthält. Nutzen Sie den  $\$$ -Operator.
2. Erstellen Sie eine Liste L2, die 20 Kubikzahlen enthält, angefangen bei der 11. Kubikzahl.
3. Hängen Sie an die Liste L2 die Zahl 3333 an (dauerhaft verändern!).
4. Fügen Sie die beiden Listen L1 und L2 zur Liste L3 zusammen und sortieren sie die Liste L3.
5. Überschreiben Sie das dritte Element mit 10 und entfernen Sie das 13. Element der Liste L3.

Lassen Sie sich jeweils die Ergebnisse (Liste und Anzahl der Elemente) anzeigen.

Speichern Sie unter **Listen.txt** ab.

## Folgen in MuPAD

- `2*n $ n=1..6;` berechnet die ersten 6 Folgenglieder der Folge  $(a_n) = 2n$
- `plot2d(Labeling=TRUE,[Mode=List,[point(n,2*n)$n=1..6]]);`  
zeichnet die ersten 6 Folgenglieder der Folge  $(a_n) = 2n$   
Es können nur Punkte gezeichnet werden.  
Die erste Koordinate  $n$  der Punkte  $P(n;2n)$  muss angegeben werden!
- `plot2d(Scaling=UnConstrained,PointWidth=50,Labeling=TRUE,[Mode=List,[point(n,2*n)$n=1..6]]);`  
Scaling = UnConstrained ... Achseneinteilung nicht vorgegeben  
Scaling = Constrained ... Achseneinteilung gleich  
PointWidth = 50 ... Dicke der gezeichneten Punkte

## Aufgaben:

1. Berechnen Sie die ersten 20 Folgenglieder und schreiben Sie die ersten 5 Folgenglieder auf.
2. Stellen Sie die Folge graphisch dar und skizzieren Sie auf dem Arbeitsblatt den Verlauf der Folge.
3. Beschreiben Sie den Verlauf der Folge verbal.

Speichern Sie unter **Folgen.txt** ab.

	Explizite Vorschrift	Folge- glieder	Skizze des Graphen	Verbale Beschreibung des Verlaufs
1	$a_n = 2n + 1$			
2	$a_n = \frac{n^2 + 1}{n + 1}$			
3	$a_n = 1 + \left(-\frac{1}{2}\right)^n$			
4	$a_n = (-1)^{n-1} \cdot \frac{12}{n}$			
5	$a_n = \frac{2n + 1}{2n - 1}$			

## Schleifen

Ein wichtiges Element der von MuPAD zur Verfügung gestellten Programmiersprache sind sogenannte Schleifen. Eine Schleife ermöglicht die wiederholte Ausführung von Befehlen.

### Die FOR-Schleife

Die folgende Eingabezeile bedeutet:

**Für i von 1 bis 4 tue folgendes:** „Schreibe den Wert von i auf den Bildschirm.“ **Ende** der Schleife

- **for i from 1 to 4 do print(i); end\_for;**

Da die Laufvariable i in ganzen Schritten hochgezählt wird, müssten die Zahlen 1, 2, 3 und 4 auf dem Bildschirm erscheinen, weil i bei 1 startet, dann den Wert 2, dann den Wert 3, dann den Wert 4 erhält und nach dem 4. Durchlauf die Schleife beendet wird.

- **for i from 1 to 5 do x:=i^2; end\_for;**

Hier wird nur 25 ausgegeben, weil die einzelnen Berechnungen der Schleife nicht auf den Bildschirm geschrieben werden, auch wenn wie hier geschehen, die Befehle mit Semikolon abgeschlossen wurden. Nur der letzte Wert von x erscheint. Will man alle Werte, so muss man die Ausgabe in jedem Schleifendurchlauf durch den *print*-Befehl erzwingen:

- **for i from 1 to 5 do x:=i^2; print(x); end\_for;**

Die folgende Schleife zählt rückwärts:

- **for i from 10 downto 3 do print(i); end\_for;**

Will man nicht in Einerschritten hoch- bzw. runterzählen, so verwende man *step*-Befehl:

- **for i from 3 to 20 step 2 do print(i); end\_for;** liefert die ungeraden Zahlen von 3 bis 20
- **for i from 100 downto 11 step 5 do print(i); end\_for;**

Dabei bricht, wie hier zu sehen, die Schleife auch ab, wenn der angegebene Endwert nicht direkt errechnet, sondern über- bzw. unterschritten wird.

Man kann auch Listen angeben, aus welcher die Laufvariable Werte annehmen soll:

- **for i in [5, 11, -14, 1.3, y, -z^3] do print(i^2); end\_for;** Quadrate der Elemente

Die Laufvariable der *FOR*-Schleife wird also in festgelegten Schritten von einem Anfangswert hoch- bzw. runtergezählt, bis ein angegebener Endwert über- bzw. unterschritten wird; oder die Laufvariable durchläuft alle Elemente einer Liste.

Eine flexiblere Schleife ist die *REPEAT*-Schleife, bei der in jedem Schritt die Laufvariable in beliebiger Weise abgeändert werden kann:

### Die REPEAT-Schleife

- **i:=2: repeat i:=i^2; print(i) until i>100 end\_repeat;** beachten Sie den “:” nach **i:=2:**

Diese Schreibweise ist etwas unübersichtlich. Nutzen Sie *SHIFT* + *ENTER*, um die Befehle untereinander zu schreiben, ohne die einzelnen Befehle sofort auszuführen.

- **i:=2:  
repeat  
    i:=i^2; print(i)  
until i>100 end\_repeat;** Mit der *TAB*-Taste (links neben Q) können sie Befehle einrücken.

Schließen Sie nun mit *ENTER* ab, so werden die Befehle zusammengefasst ausgeführt. (Ausgabe: 4, 16, 256)

*Was bedeutet diese Anweisung nun?* Im Vorfeld wird die Laufvariable auf 2 gesetzt, damit MuPAD den Anfangswert der Laufvariablen kennt. Der Doppelpunkt bedeutet, dass das Ergebnis dieses Befehls nicht auf dem Bildschirm erscheint. Ein Semikolon würde die zusätzliche Ausgabe von 2 zur Folge haben. Alles, was zwischen *repeat* und *end\_repeat* steht, wird solange wiederholt, bis die Abbruchbedingung erfüllt ist. Für dieses Beispiel heißt das: Wiederhole die Befehle **i:=i^2;** und **print(i)**, bis die Laufvariable i größer als 100 ist. Im ersten Durchlauf ist i am Anfang 2, wird dann aber sofort durch **i:=i^2** auf 4 gesetzt (Das neue i ergibt sich aus dem alten i, indem man das alte i quadriert.). Die 4 wird nun durch **print(i)** ausgegeben. Da 4 noch nicht größer als 100 ist, wird die Schleife erneut durchlaufen. Die Variable i erhält also den Wert  $4^2 = 16$ ; dieser Wert wird ausgegeben. Da 16 immer noch nicht größer als 100 ist, wird die Schleife noch mal durchlaufen; i erhält den Wert  $16^2 = 256$ , dieser Wert wird ausgegeben. Da nun aber 256 größer als 100 ist und damit die hinter **until** angegebene Abbruchbedingung erfüllt ist, wird die Schleife verlassen.

Will man neben den neuen i-Werten auch noch die alten bei jedem Schleifendurchlauf ausgegeben haben, so könnte man einen weiteren *print*-Befehl einfügen:

- **i:=2:  
repeat  
    print(i); i:=i^2; print(i)  
until i>100 end\_repeat;**

Man erhält die Ausgabe 2, 4, 4, 16, 16, 256. Die erste, dritte und fünfte Zahl sind jeweils die alten Werte.

Das sieht nicht so schön aus! Versuchen wir es etwas anders:

- **neuerWert:=2:**  
**repeat**  
     **alterWert := neuerWert; neuerWert := alterWert^2; print(alterWert, neuerWert)**  
     **until neuerWert > 100 end\_repeat;**

Hier werden jeweils die alten Werte und neuen Werte ausgegeben. Die Ausgabe der beiden Werte, getrennt durch ein Komma, ermöglicht **print(alterWert, neuerWert)**.

Am Anfang ist **neuerWert** auf 2 gesetzt. Beim 1. Durchlauf wird zuerst **alterWert** auf 2 gesetzt, dann ergibt sich der neue Wert durch Quadrieren des alten Wertes, also **neuerWert := 2<sup>2</sup> = 4**. Da 4 kleiner als 100 ist, wird die Schleife erneut durchlaufen. Der alte Wert wird durch den neuen Wert ersetzt, also **alterWert := 4**, der neue Wert entsteht durch Quadrieren des alten Wertes, also **neuerWert := 4<sup>2</sup> = 16**, usw.

Versuchen wir, wie weiter oben mit der **FOR**-Schleife, alle ungeraden Zahlen von 3 bis 20 auszugeben:

- **i:=1:**  
**repeat**  
     **i := i + 2; print(i)**  
     **until i >= 19 end\_repeat;**

Beachten Sie, dass die Schleife erst durchlaufen wird und dann die Abbruchbedingung geprüft wird. Geben Sie statt 19 den Endwert 20 an, so wird auch noch die Zahl 21 ausgegeben.

Alternativ kann man mit dem Startwert  $i = 3$  beginnen. Dann muss aber der alte  $i$ -Wert ausgegeben werden.

- **i:=3:**  
**repeat**  
     **print(i); i := i + 2**  
     **until i >= 19 end\_repeat;**

Die **REPEAT**-Schleife wiederholt also die angegebenen Befehle solange, bis die Abbruchbedingung erfüllt ist. Die Abbruchbedingung wird am Ende geprüft. Die **WHILE**-Schleife prüft am Anfang eine Bedingung und die Schleife wird nur durchlaufen, wenn die Bedingung erfüllt ist:

### Die While-Schleife

- **n := 2:**  
**while n < 100 do**  
     **a := n; n := a^2; print(a, n)**  
     **end\_while;**

Dies bedeutet: Solange die Bedingung  $n < 100$  erfüllt ist, wiederhole die Befehle. Vergleichen Sie mit der weiter oben aufgeführten **REPEAT**-Schleife. Die Variablen heißen nun „a“ und „n“, statt „alterWert“ und „neuerWert“.

### Syntax der Schleifen im Vergleich

- **for** Variable **from** Startwert **to** Endwert **step** Schrittweite **do**  
     Anweisungen  
     **end\_for;**

Von Startwert bis zum Endwert führe die Anweisungen aus. Erhöhe (erniedrige) dabei die Laufvariable um die Schrittweite.

- **Laufvariable :=** Startwert;  
**repeat**  
     Anweisungen  
     **until** Abbruchbedingung **end\_repeat;**

Wiederhole die Anweisungen solange, bis die Abbruchbedingung erfüllt ist.

(Die Schleife wird mindestens einmal durchlaufen, da die Abbruchbedingung am Ende geprüft wird.)

- **Laufvariable :=** Startwert;  
**while** Bedingung **do**  
     Anweisungen  
     **end\_while;**

Solange die Bedingung erfüllt ist, führe die Anweisungen aus. (Die Bedingung wird am Anfang geprüft.)

### Aufgaben:

1. Geben Sie die Folgenglieder 5, 11, 17, 23, 29, ..., 89 aus. Versuchen Sie dies mit allen drei Schleifen.
2. Geben Sie die Folgenglieder 89, 83, 77, 71, 65, ... 5 aus. Versuchen Sie dies mit allen drei Schleifen.
3. Berechnen Sie die Summe der Zahlen 4, 5, 6, ..., 47. Versuchen Sie dies mit allen drei Schleifen.
4. Berechnen Sie die Summe aller ungeraden Zahlen von 3 bis 29. Versuchen Sie dies mit allen drei Schleifen.

Speichern Sie unter **Schleifen.txt** ab.

## Verzweigungen

Neben den Schleifen spielen Verzweigungen beim Programmieren eine wesentliche Rolle. Je nach Wert oder Bedeutung von Variablen können durch Verzweigungen unterschiedliche Befehle ausgeführt werden. Als einfachste Variante stellt MuPAD ein *if*-Konstrukt zur Verfügung.

### Die IF – THEN – Anweisung

- **x:=2:** (Zeilenumbruch mit *Schift + Enter*)  
**if x = 2 then print("x ist 2") else print("x ist ungleich 2") end\_if;**

Diese zweite Eingabezeile bedeutet:

*Wenn* x = 2 ist, *dann* schreibe „x ist 2“ *ansonsten* schreibe „x ist ungleich 2“ *Ende* der *if-then*-Anweisung  
Als Ergebnis muss hier also „x ist 2“ erscheinen, da wir ja in der ersten Befehlszeile x mit 2 belegt haben.

- **x:=3: if x = 2 then print("x ist 2") else print("x ist ungleich 2") end\_if;**

Hier wird nun „x ist ungleich 2“ ausgegeben, da x mit 3 belegt wurde und 3 ja bekanntlich ungleich 2 ist.  
Nutzen wir eine *for*-Schleife, um alle geraden Zahlen von 1 bis 20 auszugeben:

- **for i from 1 to 20 do**  
**if testtype(i, Type::Even) then print(i) end\_if**  
**end\_for;**

Diese Anweisung bedeutet:

Für i von 1 bis 20 tue folgendes: *Wenn* i gerade ist, *dann* schreibe i auf den Bildschirm.

## Prozeduren

Mit Hilfe von Prozeduren können mehrere Anweisungen zusammengefasst werden. Prozeduren sind im wesentlichen nichts anderes als Funktionen, die bestimmte Werte zurückliefern.

Kleiden wir unser Beispiel von oben, dass testet, ob eine Zahl gleich oder ungleich 2 ist, in eine Prozedur ein:

- **GleichZwei := proc(x) # Kommentar: Test auf Gleichheit mit 2 #**  
**begin**  
**if x = 2 then return("x ist 2") else return("x ist ungleich 2") end\_if;**  
**end\_proc;**

Mit *Prozedurname:=proc(Variablen)* werden der Name und die Variablen, die beim Aufruf der Prozedur benötigt werden, festgelegt. Die Anweisungen einer Prozedur werden zwischen *begin* und *end\_proc* geschrieben. Der von uns gewählte Name für die Prozedur ist *GleichZwei*. Beim Aufruf der Prozedur wird eine Variable (eine Zahl x) erwartet. Innerhalb der Prozedur steht nun die von oben bekannte *if-then*-Anweisung. Der Befehl *print* wurden durch *return* ersetzt. Bei der Abarbeitung wird die Prozedur durch *return* beendet und der bei *return* angegebene Wert wird ausgegeben. Schließt man die Prozedur mit einem Doppelpunkt hinter *end\_proc* ab, so erscheint der Prozedurtext nicht auf dem Bildschirm.

Der Aufruf der Prozedur besteht aus dem Prozedurnamen, gefolgt von den benötigten Zahlen:

- **GleichZwei(4); GleichZwei(2);**

Die folgende Prozedur liefert das Maximum zweier Zahlen zurück:

- **Maximum := proc(a, b) # Kommentar: das Maximum von a und b #**  
**begin**  
**if a < b then return(b) else return(a) end\_if**  
**end\_proc:**

Beim Aufruf der Prozedur werden 2 Variablen (zwei Zahlen a und b) erwartet. Innerhalb der Prozedur steht eine *if-then*-Anweisung, die die beiden Zahlen a und b vergleicht und jeweils das Maximum der Zahlen zurückgibt.

- **Maximum(4, 7); Maximum(-6, 3); Maximum(3/7, 111/212);**

## Aufgaben

1. Untersuchen Sie, was die einzelnen Befehle der folgenden Prozedur bewirken und testen sie die Prozedur.

- **GeradeZahlen := proc(Anfang, Ende) local i, Liste;**  
**begin**  
**Liste := [ ];**  
**for i from Anfang to Ende do**  
**if testtype(i, Type::Even) then Liste := append(Liste, i) end\_if**  
**end\_for;**  
**return(Liste);**  
**end\_proc:**

(*local i, Liste;* legt lokale Variablen fest. Betrachten Sie i und Liste, wie sonst, als normale Variablen.)

2. Schreiben Sie eine Prozedur, die alle negativen ganzen Zahlen einer Eingabe in eine Liste schreibt und diese ausgibt. Eingabe der Prozedur sollen wieder ein Anfangswert und ein Endwert sein (Diese Werte sollen angeben, welche ganzen Zahlen untersucht werden sollen.). Nutzen Sie *Type::NegInt*.

Speichern Sie ihre Arbeit unter *proz1.txt* ab und **drucken** Sie die Prozedur von Aufgabe 2 eventuell **aus**.

## Erste Prozeduren zur Zahlentheorie

### 1. Teiler einer natürlichen Zahl $n$

- a) Erstellen Sie eine Prozedur **Teiler**, die alle positiven Teiler einer Zahl in eine Liste schreibt.  
*Hinweise:* Eingabe: positive Zahl  $n$   
Ausgabe: Liste, die alle positiven Teiler von  $n$  enthält  
Befehle: **irem( $n, m$ )** Rest bei der ganzzahligen Division von  $n$  durch  $m$   
Beispiel: **irem(16, 3);** Ausgabe: **1** da  $16 = 5 \cdot 3 + 1$
- b) Erstellen Sie eine Prozedur **TeilerAnzahl**, die die Anzahl der positiven Teiler einer Zahl berechnet.  
*Hinweise:* Eingabe: positive Zahl  $n$   
Ausgabe: Anzahl der positiven Teiler von  $n$   
Befehle: **nops(Liste)** Anzahl der Elemente einer Liste  
 Nutzen Sie die Prozedur **Teiler**, indem Sie die Anzahl der von der Prozedur Teiler zurückgelieferten Liste berechnen lassen.

### 2. Teilerfremde Zahlen zu einer natürlichen Zahl $n$

- a) Erstellen Sie eine Prozedur **Teilerfremd**, die alle zu  $n$  teilerfremden Zahlen in eine Liste schreibt.  
*Hinweise:* Eingabe: positive Zahl  $n$   
Ausgabe: Liste, die alle zu  $n$  teilerfremden Zahlen enthält  
Befehle: **igcd( $n, m$ )** größter gemeinsamer Teiler von  $n$  und  $m$   
Beispiel: **igcd(16, 12);** Ausgabe: 4  
 Zwei Zahlen heißen teilerfremd  $\Leftrightarrow$  der größte gemeinsame Teiler ist ...
- b) Erstellen Sie eine Prozedur **TeilerfremdAnzahl**, die die Anzahl der teilerfremden Zahlen berechnet.  
*Hinweise:* Eingabe: positive Zahl  $n$   
Ausgabe: Anzahl der zu  $n$  teilerfremden Zahlen

### 3. Primteiler zu einer natürlichen Zahl $n$

- a) Erstellen Sie eine Prozedur **Primteiler**, die alle Primteiler von  $n$  in eine Liste schreibt.  
*Hinweise:* Eingabe: positive Zahl  $n$   
Ausgabe: Liste, die alle Primteiler von  $n$  enthält  
Befehle: **igcd( $n, m$ )** größter gemeinsamer Teiler von  $n$  und  $m$   
**isprime( $n$ )** **TRUE**  $\Leftrightarrow$   $n$  Primzahl, **FALSE** sonst
- Erstellen Sie eine Prozedur **PrimteilerAnzahl**, die die Anzahl der Primteiler von  $n$  berechnet.  
*Hinweise:* Eingabe: positive Zahl  $n$   
Ausgabe: Anzahl der Primteiler von  $n$

### 4. Primzahlen

- a) Erstellen Sie eine Prozedur **Primzahlen**, die alle Primzahlen in einem abgeschlossenen Intervall in eine Liste schreibt.  
*Hinweise:* Eingabe: Anfang und Ende des Intervalls  
Ausgabe: Liste, die alle Primzahlen des Intervalls enthält  
Befehle: **isprime( $n$ )**
- b) Erstellen Sie eine Prozedur **PrimzahlenAnzahl**, die die Anzahl der Primzahlen eines Intervalls berechnet.  
*Hinweise:* Eingabe: Anfang und Ende des Intervalls  
Ausgabe: Anzahl der Primzahlen im Intervall

### 5. Zusammengesetzte Zahlen

- a) Erstellen Sie eine Prozedur **ZusammengesetztZahlen**, die alle zusammengesetzten Zahlen in einem abgeschlossenen Intervall in eine Liste schreibt.  
*Hinweise:* Eingabe: Anfang und Ende des Intervalls  
Ausgabe: Liste, die alle zusammengesetzten Zahlen des Intervalls enthält  
Befehle: **isprime( $n$ )**
- b) Erstellen Sie eine Prozedur **ZusammengesetztZahlenAnzahl**, die die Anzahl der zusammengesetzten Zahlen eines Intervalls berechnet.  
*Hinweise:* Eingabe: Anfang und Ende des Intervalls  
Ausgabe: Anzahl der zusammengesetzten Zahlen im Intervall

## Zahlentheorie: Grundlagen

$a, b \in \mathbb{Z}$ . Eine Zahl  $a$  heißt **Teiler von einer Zahl  $b$** , wenn eine Zahl  $m \in \mathbb{Z}$  existiert, so dass gilt:  $b = m \cdot a$ .

Schreibweise:  $a|b$  Beispiele:  $7|21$ , da  $21 = 3 \cdot 7$ ;  $12|-60$ , da  $-60 = -5 \cdot 12$ ;  $-6|24$ , da  $24 = -4 \cdot (-6)$

**Teilbarkeitseigenschaften** Für alle  $a, b, c, d, m, n \in \mathbb{Z}$  gilt:

Nr.	Eigenschaft	Beispiele	Erläuterung und Beweisskizze
1.	$a a$	$6 6; -12 -12$	Jede Zahl teilt sich selbst, da gilt: $a = 1 \cdot a$
2.	$a 0$	$14 0; -7 0$	Jede Zahl teilt die Null, da gilt: $0 = 0 \cdot a$
3.	$1 a$	$1 15; 1 -3$	Eins teilt jede Zahl, da gilt: $a = a \cdot 1$
4.	$a b$ und $b a \Rightarrow a = \pm b$	$6 6$ und $6 6 \Rightarrow 6 = 6$ $6 -6$ und $-6 6$ $\Rightarrow 6 = -(-6)$	Wenn $a$ die Zahl $b$ teilt und $b$ gleichzeitig $a$ teilt, dann müssen die Beträge von $a$ und $b$ gleich sein: $b = m \cdot a$ und $a = n \cdot b \Rightarrow a = (n \cdot m) \cdot a$ $\Rightarrow n \cdot m = 1 \Rightarrow (n=1 \wedge m=1) \vee (n=-1 \wedge m=-1) \Rightarrow a = \pm b$
5.	$a b$ und $b c \Rightarrow a c$	$3 9$ und $9 27 \Rightarrow 3 27$ $3 9 \Rightarrow 3 18; 3 27; 3 36$ $\Rightarrow 3 -9; 3 -18; 3 -27$	Wenn $a$ ein Teiler von $b$ ist, dann teilt $a$ auch jede Zahl, die $b$ teilt: $b = m \cdot a$ und $c = n \cdot b \Rightarrow c = (n \cdot m) \cdot a \Rightarrow a c$
6.	$a b$ und $a c \Rightarrow a (b \pm c)$	$4 8$ und $4 20$ $\Rightarrow 4 28; 4 -12$	Wenn $a$ sowohl $b$ als auch $c$ teilt, so teilt $a$ auch die Summe und die Differenz von $b$ und $c$ : $b = m \cdot a$ und $c = n \cdot a \Rightarrow b \pm c = (m \pm n) \cdot a$ $\Rightarrow a (b \pm c)$
7.	$a b$ und $c d \Rightarrow a \cdot c   b \cdot d$	$5 10$ und $3 9 \Rightarrow 15 90$ $-3 9$ und $2 6 \Rightarrow -6 54$	Das Produkt der Teiler zweier Zahlen $b$ und $d$ teilt das Produkt der zwei Zahlen $b$ und $d$ : $b = m \cdot a$ und $d = n \cdot c \Rightarrow b \cdot d = (m \cdot n) \cdot a \cdot c$ $\Rightarrow a \cdot c   b \cdot d$
8.	$a b \Rightarrow a   b \cdot c$	$7 14 \Rightarrow 7 28$ $\Rightarrow 7 -56$	Ist $a$ ein Teiler von $b$ , so teilt $a$ auch jedes Vielfache von $b$ : Regel 7 mit $a b$ und $1 c$

Jede natürliche Zahl, die größer 1 ist und die nur durch 1 und sich selbst teilbar ist, heißt **Primzahl**.

*Folgerung:* Eine Primzahl  $p$  lässt sich nicht als Produkt  $p = p_1 \cdot p_2$  mit  $1 < p_1$  und  $1 < p_2$  ( $p_1, p_2 \in \mathbb{N}$ ) schreiben.

Eine natürliche Zahl, die größer als 1 ist und außer 1 und sich selbst noch mindestens einen weiteren positiven Teiler besitzt, heißt **zusammengesetzte Zahl**.

*Folgerung:* Eine zusammengesetzte Zahl  $n$  lässt sich als Produkt  $n = n_1 \cdot n_2$  mit  $1 < n_1$  und  $1 < n_2$  ( $n_1, n_2 \in \mathbb{N}$ ) schreiben.

Eine Zahl, die Teiler mehrerer Zahlen ist, heißt **gemeinsamer Teiler** dieser Zahlen.

*Beispiel:* gemeinsame Teiler von 28 und 42: 1, 2, 7, 14

Der **größte gemeinsame Teiler** gegebener Zahlen ist die größte Zahl, die alle diese Zahlen teilt.

*Schreibweise:*  $\text{ggT}(a, b, c, \dots)$  *Beispiel:*  $\text{ggT}(28, 42) = 14$

Zahlen, die außer 1 keinen gemeinsamen Teiler haben, heißen zueinander **teilerfremd**.

*Beispiel:* 21 und 40 sind teilerfremd *Achtung:* 1 ist Teiler jeder Zahl, aber 1 ist zu jeder Zahl teilerfremd

*Folgerung:* Zwei natürliche Zahlen  $a$  und  $b$  sind teilerfremd  $\Leftrightarrow \text{ggT}(a, b) = 1$ .

Zwei natürliche Zahlen  $m$  und  $n$  sind nicht teilerfremd  $\Leftrightarrow \text{ggT}(m, n) > 1$ .

$p$  Primzahl,  $a \in \mathbb{N}$ :  $\text{ggT}(a, p) = 1 \Leftrightarrow a$  und  $p$  teilerfremd  $\Leftrightarrow a$  kein Vielfaches von  $p$

Ein Teiler einer Zahl heißt **Primteiler** der Zahl, wenn der Teiler eine Primzahl ist.

*Beispiel:* 2, 3 und 7 sind Primteiler von 42

**Satz:** Jede natürliche Zahl  $n > 1$  ist Primzahl oder lässt sich **eindeutig** als Produkt von Primzahlen darstellen.

*Beispiele:*  $28 = 2 \cdot 2 \cdot 7$ ;  $42 = 2 \cdot 3 \cdot 7$

*Folgerung:* Jede natürliche Zahl  $n > 1$  besitzt mindestens einen Primteiler.

## Primzahlen – damals und heute

### Fermat in einem Brief an Mersenne im Dezember 1640:

([http://www.scruznet.com/~luke/lit/lit\\_068s.htm](http://www.scruznet.com/~luke/lit/lit_068s.htm))

*"If I can determine the basic reason why  
3, 5, 7, 17, 257, 65 537, ...,  
are prime numbers, I feel that I would find very  
interesting results, for I have already found marvelous  
things which I will tell you about later."*



### PrimeNet's discovery notification message:

(<http://mersenne.org/ips/index.html>)

From: primenet@entropia.com [mailto:primenet@entropia.com]  
Sent: Tuesday, June 01, 1999 5:57 AM  
To: woltman@magicnet.net; primenet@entropia.com; notification\_list  
Subject: M6972593 Reported Prime on GIMPS PrimeNet  
-----Message-----

Advance notice sent by Internet PrimeNet Server 4.0.017

----- UNVERIFIED -----

M6972593 Reported Prime on GIMPS PrimeNet

-----

Exponent : 6972593  
Time stamp, UTC : 99/06/01 13:57:26  
Lucas-Lehmer residue : 0x0000000000000000  
No factor thru bits : 62.0  
Account PrimeNet ID : nayan  
Account computer ID : precision-mm  
Account e-mail : nayan@walanet.com

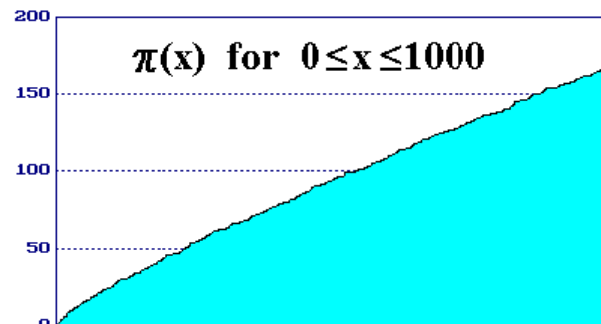
----- UNVERIFIED -----

Entropia.com, Inc.  
Internet Research Computing Technologies  
info@entropia.com  
+1 408 807-7003  
(c)1997-1999

## Wie viele Primzahlen gibt es?

### Anzahl der Primzahlen in gleich großen Intervallen

Intervallgrenzen	Anzahl der Primzahlen im Intervall
1 und 10	4
10 und 20	4
20 und 30	2
30 und 40	2
90 und 100	1
100 und 110	4
320 und 330	0
820 und 830	4



### Anzahl der Primzahlen bis zu einer Grenze

$x$	Anzahl der Primzahlen kleiner gleich $x$	Anteil der Primzahlen
10	4	40,0%
100	25	
1,000	168	
10,000	1,229	
100,000	9,592	
1,000,000	78,498	
10,000,000	664,579	
100,000,000	5,761,455	
1,000,000,000	50,847,534	
10,000,000,000	455,052,511	
100,000,000,000	4,118,054,813	
1,000,000,000,000	37,607,912,018	
10,000,000,000,000	346,065,536,839	
100,000,000,000,000	3,204,941,750,802	
1,000,000,000,000,000	29,844,570,422,669	
10,000,000,000,000,000	279,238,341,033,925	
100,000,000,000,000,000	2,623,557,157,654,233	
1,000,000,000,000,000,000	24,739,954,287,740,860	
10,000,000,000,000,000,000	234,057,667,276,344,607	
100,000,000,000,000,000,000	2,220,819,602,560,918,840	
1,000,000,000,000,000,000,000	21,127,269,486,018,731,928	
10,000,000,000,000,000,000,000	201,467,286,689,315,906,290	

**Verschiedene Leute sollen beweisen:****Alle ungeraden Zahlen größer Eins sind Primzahlen.**

Mathematiker:	3 ist prim; 5 ist prim; 7 ist prim; 9 ist nicht prim ⇒ Gegenbeispiel gefunden ⇒ Die Behauptung ist falsch.
Physiker:	3 ist prim; 5 ist prim; 7 ist prim; 9 ist ein Messfehler; 11 ist prim,...
Biologe:	3 ist prim; 5 ist prim; 7 ist prim; 9 ist prim; 11 ist prim;...
Jurist:	3 ist prim, da haben wir ja schon den Präzedenzfall
Politiker:	3 ist prim; 5 ist prim; 7 ist prim; 9 ist in der Minderheit, können wir ignorieren; 11 ist prim; 13 ist prim;...
Psychologe:	3 ist prim; 5 ist prim; 7 ist prim; 9 ist prim, aber unterdrückt es; 11 ist prim; 13 ist prim;...
Windows Benutzer:	3 ist prim; 5 ist prim; 7 ist prim; 9 ist... - Allgemeine Schutzverletzung im Modul primzahl.dll
Informatiker:	3 ist prim; 5 ist prim; 7 ist prim; 7 ist prim; 7 ist prim; 7 ist prim; 7 ist prim; 7 ist prim;... - STACK OVERFLOW
Soziologe:	3 ist eine Zahl, 3 ist eine Primzahl - Alle Zahlen sind Primzahlen.
Professor:	3 ist prim. Der Rest wird dem Studenten als triviale Übungsaufgabe überlassen.

## Fermatsche und Mersennesche Zahlen

### Hausaufgabe:

- Testen Sie die ersten 11 Fermatschen Zahlen und die ersten 21 Mersenneschen Zahlen auf die Primzahleigenschaft.
- Was fällt Ihnen bei den Mersenneschen Primzahlen auf?

**Befehle:**    `isprime(n)`                    `TRUE`  $\Leftrightarrow$  Primzahl; `FALSE`  $\Leftrightarrow$  keine Primzahl  
                   `evtl.: Folgen`                    `beachte: (2^(2^n)+1)`

**Zusatz:**    Finden Sie die Primfaktoren der zusammengesetzten Zahlen. *Befehl:* `ifactor(n)`;

n	$F_n = 2^{2^n} + 1$	Primzahl?	$M_n = 2^n - 1$	Primzahl?
0	3	TRUE	-1	FALSE
1	5	TRUE	1	FALSE
2	17		3	
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

## Fermatsche Zahlen

**Fermat (1601 – 1655): Alle Zahlen der Form  $F_n = 2^{2^n} + 1$  sind Primzahlen.**

- $F_5 = 4294967297 \Rightarrow$  Kannte Fermat *alle* Primzahlen bis **65536**?
- $F_9$  und  $F_{11}$ : größte Fermat-Zahlen, von denen man die *komplette Primzahlzerlegung* kennt.  
( $F_{11}$  vor  $F_9$  faktorisiert, weil  $F_{11}$  zwei kleine Primfaktoren besitzt)
- $F_4 = 65537$ : *größte bekannte* Fermatsche Primzahl
- rund **135 zusammengesetzte** Fermat-Zahlen bekannt
- $F_{23471}$ : *größte bekannte* zusammengesetzte Fermat-Zahl  
(mehr als  $10^{7000}$  Stellen) (Keller fand 1984 einen Faktor:  $5 \cdot 2^{23473} + 1$ )
- $F_{24}, F_{28}$ : kleinste Fermat-Zahlen, von denen *nicht bekannt* ist, ob sie Primzahlen oder zusammengesetzt sind

### Primfaktoren der zusammengesetzten Fermat-Zahlen bis $F_{11}$

Primfaktorzerlegung von $F_n$	Entdeckung
$F_5 = 641 \cdot 6700417$	Euler 1732
$F_6 = 274177 \cdot 67280421310721$	Clausen 1855 (Erwähnung in Brief an Gauß vom 01.01.1856) Nachweis 1880 durch Landry und Le Lasseur
$F_7 = 59649589127497217 \cdot 5704689200685129054721$	Morrison und Brillhart 1970 (veröffentlicht: 1975)
$F_8 = 1238926361552897 \cdot 9346163971535797769163558199606896584051237541638188580280321$	Brent und Pollard 1981
$F_9 = 2424833 \cdot 7455602825647884208337395736200454918783366342657 \cdot 741640062627530801524787141901937474059940781097519023905821316144415759504705008092818711693940737$	A. Lenstra und Manasse 1990 (700 vernetzte PC's weltweit und ein Supercomputer; Rechenzeit: 4 Monate)
$F_{10} = 45592577 \cdot 6487031809 \cdot ?$	Selfridge 1953
$F_{11} = 319489 \cdot 974849 \cdot 167988556341760475137 \cdot 3560841906445833920513 \cdot P$ (P...Primzahl mit über 564 Stellen)	Brent 1989 (Zerlegung) Morain 1989 (Nachweis, dass P eine Primzahl ist)

## Mersennesche Zahlen

- Antike bis Mittelalter: Für jede Primzahl  $p$  ist  $2^p - 1$  eine Primzahl.
- 1456: unbekannter Mathematiker:  $2^{13} - 1$  ist prim
- 1536: Hudalricus Regius:  $2^{11} - 1 = 2047 = 23 \cdot 89$
- 1603: Pietro Cataldi:  $2^{17} - 1$  und  $2^{19} - 1$  sind prim  
Behauptung:  $2^{23} - 1; 2^{29} - 1; 2^{31} - 1; 2^{37} - 1$  sind prim
- 1640: Pierre Fermat:  $2^{23} - 1$  und  $2^{37} - 1$  zusammengesetzt
- 1738: Leonhard Euler:  $2^{29} - 1$  zusammengesetzt
- 1750: Leonhard Euler:  $2^{31} - 1$  prim
- 1876: Lucas:  $2^{127} - 1$  ist prim
- **1644: Marin Mersenne (franz. Mönch 1588 - 1648):**

**Behauptung: Für alle Primzahlen bis 257 liefern nur die Fälle**

$$\mathbf{p = 2, 3, 5, 7, 13, 17, 19, 31, 67, 127, 257}$$

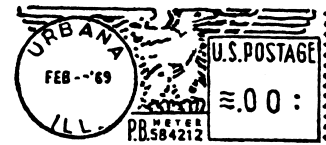
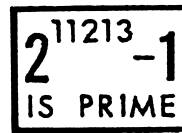
### **Primzahlen.**

- 1883: Pervouchine:  $p = 61$  übersehen
- 1911: Powers:  $p = 89$  übersehen
- 1914: Powers:  $p = 107$  übersehen
- 1947: endgültige Überprüfung bis  $p = 257$  abgeschlossen

$\Rightarrow M_p = 2^p - 1$  sind Primzahlen für

$$\mathbf{p = 2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127}$$

- 1963: Gillies:  $2^{11213} - 1$  ist prim



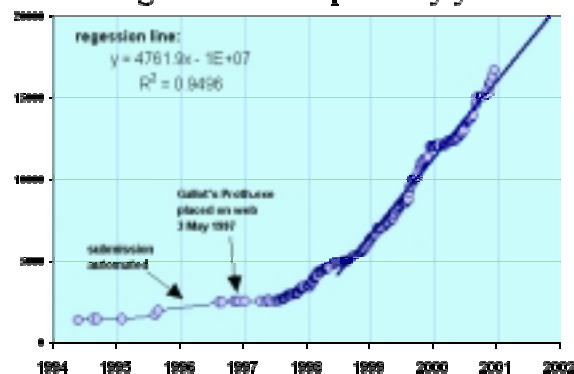
### Primfaktoren der zusammengesetzten Mersenneschen Zahlen bis $M_{20}$

Primfaktorzerlegung von $M_n$
$M_4 = 15 = 3 \cdot 5$
$M_6 = 63 = 3 \cdot 3 \cdot 7$
$M_8 = 255 = 3 \cdot 5 \cdot 17$
$M_9 = 511 = 7 \cdot 73$
$M_{10} = 1023 = 3 \cdot 11 \cdot 31$
$M_{11} = 2047 = 23 \cdot 89$
$M_{12} = 4095 = 3 \cdot 3 \cdot 5 \cdot 7 \cdot 13$
$M_{14} = 16383 = 3 \cdot 43 \cdot 127$
$M_{15} = 32767 = 7 \cdot 31 \cdot 151$
$M_{16} = 65535 = 3 \cdot 5 \cdot 17 \cdot 257$
$M_{18} = 262143 = 3 \cdot 3 \cdot 3 \cdot 7 \cdot 19 \cdot 73$
$M_{20} = 1048575 = 3 \cdot 5 \cdot 5 \cdot 11 \cdot 31 \cdot 41$

Primzahlentdeckungen, bevor es Computer gab:  
[http://www.utm.edu/research/primes/notes/by\\_year.html#1](http://www.utm.edu/research/primes/notes/by_year.html#1)

Größte bekannte Primzahlen:  
<http://www.utm.edu/research/primes/largest.html>

### Digits in 5000<sup>th</sup> prime by year



### Größte bekannte Primzahl: $2^{6972593} - 1$ (30.06.1999)

<http://www.utm.edu/research/primes/notes/6972593/>

- erste bekannte Primzahl mit über 1Mio Stellen (2098960 Stellen)
- Computer: 350 MHz IBM Aptiva
  - Testdauer: 111 Tage (ohne Unterbrechung: 3 Wochen)
  - Testdauer mit einem besseren Testprogramm: 2 Wochen (500 MHz Alpha workstation)
- \$50000 für Entdeckung
- ausgeschrieben im Textfile: 2MB
- Länge der Primzahl: <http://www.utm.edu/research/primes/cgi/HowLong.cgi/6972593/>

### PrimeNet

- Koordination von
  - mehr als 21500 Rechnern
  - mehr als 12600 Internetbenutzern
  - 720 Billionen Berechnungen pro Sekunde
- Preisgeld: **\$100000** für erste Primzahl mit mehr als 10Mio Stellen (nicht viel schwerer als Berechnung der Primzahl mit 2Mio Stellen)
- Internetadresse: Links zu Primzahlen siehe [www.bernheiden.de](http://www.bernheiden.de)

## Erste Primzahltests

1. Erstellen Sie eine Prozedur *isprime1*, die den folgenden Primzahltest benutzt:

**Test 1:**  $n$  Primzahl  $\Leftrightarrow$  Für alle  $a \in \mathbb{N}$  mit  $1 < a < n$  gilt:  $\text{ggT}(a, n) = 1$ .  
**Eingabe:** positive Zahl  $n$   
**Ausgabe:**  $n$  Primzahl  $\Rightarrow$  **TRUE**;  $n$  keine Primzahl  $\Rightarrow$  **FALSE**  
**Befehle:** **igcd(n, m)**  $\text{ggT}(n, m)$

im Detail:

1. Ist  $n < 2$ , dann ist  $n$  keine Primzahl.
2. Wenn ein  $a$  ( $1 < a < n$ ) gefunden wird, so dass  $\text{ggT}(a, n) > 1$  ist, dann haben  $a$  und  $n$  einen gemeinsamen Teiler, der größer als 1 ist. Also besitzt  $n$  einen Teiler, der größer als 1, aber kleiner als  $n$  ist ( $a$  läuft nur von 2 bis  $n - 1$ ), die Zahl  $n$  ist demnach keine Primzahl.
3. Ist  $n \geq 2$  und ergibt sich für alle  $a$  ( $1 < a < n$ ) immer  $\text{ggT}(a, n) = 1$ , so besitzt  $n$  keine weiteren Teiler als 1 und sich selbst. Also ist  $n$  eine Primzahl.

in MuPAD:

1. **if-then**-Anweisung mit **return(FALSE)**
2. **for**-Schleife mit **if-then**-Anweisung mit **return(FALSE)**
3. **return(TRUE)**

Bei der Abarbeitung wird die Prozedur durch **return** beendet und der bei **return** angegebene Wert wird ausgegeben. Wird ein **return**-Befehl abgearbeitet, wird die Prozedur sofort verlassen. Weitere Anweisungen der Prozedur werden nicht mehr bearbeitet. Es reicht deshalb, nach dem Testen auf 1. und 2. nur noch **return(TRUE)** hinzuschreiben.

Die Befehle innerhalb einer **for**-Schleife werden nur abgearbeitet, wenn sich die Laufvariable innerhalb des angegebenen Bereiches bewegt. So werden die Befehle der **for**-Schleife nicht abgearbeitet, wenn *isprime1*(2) aufgerufen wird, weil die **for**-Schleife dann von 2 hochzählend bis  $2 - 1 = 1$  laufen müsste.

Frage: Warum lässt man  $a$  von 2 bis  $n - 1$  laufen? Dürfte man dies überhaupt? Begründen sie.

Antwort: \_\_\_\_\_  
 \_\_\_\_\_

2. Erstellen Sie eine Prozedur *isprime2*, die den folgenden Primzahltest benutzt:

**Test 2:**  $n$  Primzahl  $\Leftrightarrow$  Für alle  $a \in \mathbb{N}$  mit  $1 < a \leq \lfloor \sqrt{n} \rfloor$  gilt:  $\text{ggT}(a, n) = 1$ .  
**Befehle:** **trunc(sqrt(n))** größte ganze Zahl kleiner gleich  $\sqrt{n}$

3. Vergleichen Sie *isprime1* und *isprime2* bei großen Zahlen ( $n > 100000$ ). Was stellen Sie fest? Für einen Zeitvergleich können Sie den Befehl **time()** nutzen. **time()** gibt die gesamte Zeit in Millisekunden zurück, die seit Beginn der MuPAD-Sitzung vergangen ist.

z.B.: **Startzeit := time() ; isprime1(100003); (time() - Startzeit) \* ms;**

Ergebnisse: \_\_\_\_\_  
 \_\_\_\_\_  
 \_\_\_\_\_

**Zusatz 1:** Erstellen Sie eine Prozedur *Primzahlliste2*, die mit Hilfe von *isprime2* eine Primzahlliste erstellt.

**Eingabe:** Anfangswert und Endwert

**Ausgabe:** Liste, die alle Primzahlen zwischen Anfangswert und Endwert enthält

Speichern Sie die Arbeit unter *prime1.mnb* ab.

**Zusatz 2:** Informieren Sie sich über Mersennesche Primzahlen. (Linkliste unter [www.bernheiden.de](http://www.bernheiden.de))

**Aufgabe:**

Zeigen Sie: Mersennesche Zahlen  $M_p = 2^p - 1$  liefern nur Primzahlen, wenn  $p$  auch eine Primzahl ist.

Beachte:

Dass  $p$  eine Primzahl ist, bedeutet nicht notwendigerweise, dass  $M_p$  auch eine Primzahl ist: **Beispiel?**

**Satz:** ( $m, n \in \mathbb{N}$ ;  $m > 1$ ;  $n > 1$ )

Falls  $p = m \cdot n$ , so ist  $M_p = 2^p - 1$  zusammengesetzt.

Beispiel:  $p = 2 \cdot n$

$$\Rightarrow 2^p - 1 = 2^{2n} - 1 = (2^n)^2 - 1 = ? \quad (\text{Binomische Formel})$$

$$\Rightarrow 2^p - 1 \text{ lässt sich als Produkt mit zwei Faktoren darstellen, die beide ...}$$

$$\Rightarrow 2^p - 1 \text{ ist ...}$$

zum Beweis:

Schreiben Sie die  $n$ -te Partialsumme für eine geometrische Zahlenfolge mit  $a_1 = 1$  auf. Schreiben Sie die Summe ausführlich, multiplizieren Sie mit dem Nenner der rechten Seite der Gleichung und ersetzen Sie  $q$  geeignet.

**Lösung:**

Mersennesche Zahlen  $M_p = 2^p - 1$  liefern nur Primzahlen, wenn  $p$  auch eine Primzahl ist.

Beachte:

Dass  $p$  eine Primzahl ist, bedeutet nicht notwendigerweise, dass  $M_p$  auch eine Primzahl ist:  $M_{11} = 23 \cdot 89$ .

Beispiel:  $p = 2 \cdot n$

$$\Rightarrow 2^p - 1 = 2^{2n} - 1 = (2^n)^2 - 1 = (2^n - 1) \cdot (2^n + 1)$$

$$\text{Binomische Formel: } a^2 - b^2 = (a + b) \cdot (a - b)$$

$$\Rightarrow 2^p - 1 \text{ lässt sich als Produkt mit zwei Faktoren darstellen, die beide größer 1 sind.}$$

$$\Rightarrow 2^p - 1 \text{ ist zusammengesetzt.}$$

Beweis:

Partialsumme einer geometrischen Zahlenfolge mit  $q \neq 1$ :  $\sum_{k=1}^n q^{k-1} = \frac{q^n - 1}{q - 1}$

$$\Rightarrow 1 + q + q^2 + q^3 + \dots + q^{n-1} = \frac{q^n - 1}{q - 1}$$

$$\Rightarrow (1 + q + q^2 + q^3 + \dots + q^{n-1}) \cdot (q - 1) = q^n - 1$$

Ersetze  $q$  durch  $2^m$ :

$$\Rightarrow (1 + 2^m + (2^m)^2 + (2^m)^3 + \dots + (2^m)^{n-1}) \cdot (2^m - 1) = (2^m)^n - 1$$

$$\Rightarrow (1 + 2^m + 2^{2m} + 2^{3m} + \dots + 2^{(n-1)m}) \cdot (2^m - 1) = 2^{m \cdot n} - 1$$

$$\Rightarrow 2^{m \cdot n} - 1 \text{ lässt sich als Produkt mit zwei Faktoren darstellen, die beide größer 1 sind.}$$

$$\Rightarrow M_p = 2^p - 1 = 2^{m \cdot n} - 1 \text{ ist zusammengesetzt}$$

**Sieb des Eratosthenes**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

**Sieb des Ulam:**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

**Aufgaben:**

- Finden Sie alle Primzahlen von 1 bis 55 mit dem Sieb des Eratosthenes.
  - Ab welcher Zahl können Sie die übrigen alle markieren?
  - Wie viele Primzahlen haben sie gefunden?
- Finden Sie alle Primzahlen von 1 bis 210 mit dem Sieb des Ulam.
  - Ab welcher Zahl können Sie die übrigen alle markieren?
  - Wie viele Primzahlen haben Sie gefunden?
  - Vergleichen Sie die beiden Siebe.
  - Welche Gesetzmäßigkeit können Sie für alle Primzahlen größer 3 ableiten?
  - Wie viele Primzahlrillinge ( $p$ ;  $p + 2$ ;  $p + 4$ ) gibt es?

**Sieb des Eratosthenes**

Aus einer unendlich langen Liste natürlicher Zahlen lässt sich jede Primzahl nach und nach finden:

- Die 1 ist per Definition keine Primzahl, sie wird gestrichen.
- Die erste ungestrichene Zahl ist nun eine Primzahl: die 2. Sie wird markiert und alle Vielfachen der 2 werden gestrichen.
- Die nächste nicht gestrichene Zahl ist wieder eine Primzahl: die 3. Sie wird markiert und alle ihre Vielfachen werden gestrichen.
- ...

**Sieb des Ulam**

Eine unendlich lange Liste natürlicher Zahlen wird in 6 Spalten (siehe links) aufgeschrieben. Gestrichen und markiert wird wie beim Sieb des Eratosthenes.

## Rechenregeln für Kongruenzen

$a \equiv b \pmod{m} \Leftrightarrow a$  und  $b$  lassen bei Division durch  $m$  den gleichen Rest  $\Leftrightarrow m \mid (a - b)$

Für alle  $a, b, c, d, r, a_1, b_1 \in \mathbb{Z}; m \in \mathbb{N} \setminus \{1\}$  gilt:

Nr.	Rechenregel	Beispiele	Erläuterung in Kurzform und Beweisskizze
1.	$a \equiv a \pmod{m}$	$6 \equiv 6 \pmod{5}$	Jede Zahl ist zu sich selbst kongruent: $m \mid (a - a)$
2.	$a \equiv b \pmod{m} \Rightarrow b \equiv a \pmod{m}$	$6 \equiv 11 \pmod{5}$ $\Rightarrow 11 \equiv 6 \pmod{5}$	$a$ und $b$ lassen bei Division durch $m$ den gleichen Rest: $m \mid (a - b) \Rightarrow \dots \Rightarrow m \mid (b - a) \Rightarrow \dots$
3.	$a \equiv b \pmod{m}$ und $b \equiv c \pmod{m}$ $\Rightarrow a \equiv c \pmod{m}$	$13 \equiv 27 \pmod{7}$ und $27 \equiv 41 \pmod{7}$ $\Rightarrow 13 \equiv 41 \pmod{7}$	Wenn $a$ und $b$ den gleichen Rest lassen und $b$ und $c$ den gleichen Rest lassen, dann lassen auch $a$ und $c$ den gleichen Rest: $m \mid (a - b)$ und $m \mid (b - c) \Rightarrow m \mid (a - b + b - c) \Rightarrow \dots$
4.	<b>Addition/Subtraktion</b> $a \equiv b \pmod{m}$ und $c \equiv d \pmod{m}$ $\Rightarrow a \pm c \equiv b \pm d \pmod{m}$	$48 \equiv 27 \pmod{7}$ und $18 \equiv 25 \pmod{7}$ $\Rightarrow 66 \equiv 52 \pmod{7}$ $\Rightarrow 30 \equiv 2 \pmod{7}$	Kongruenzen zum gleichen Modul können addiert und subtrahiert werden: $m \mid (a - b)$ und $m \mid (c - d) \Rightarrow m \mid (a - b) \pm (c - d) \Rightarrow \dots$
5.	$a \equiv b \pmod{m}$ $\Rightarrow a \pm c \equiv b \pm c \pmod{m}$	$31 \equiv 67 \pmod{9}$ $\Rightarrow 33 \equiv 69 \pmod{9}$ $\Rightarrow 20 \equiv 56 \pmod{9}$	Wenn $a$ und $b$ bei Division durch $m$ den gleichen Rest lassen, dann auch $a \pm c$ und $b \pm c$ : siehe 4. mit $a \equiv b \pmod{m}$ und $c \equiv c \pmod{m}$
6.	<b>Multiplikation</b> $a \equiv b \pmod{m}$ und $c \equiv d \pmod{m}$ $\Rightarrow a \cdot c \equiv b \cdot d \pmod{m}$	$15 \equiv 39 \pmod{6}$ und $4 \equiv 16 \pmod{6}$ $\Rightarrow 60 \equiv 624 \pmod{6}$	Kongruenzen zum gleichen Modul können multipliziert werden: $m \mid (a - b)$ und $m \mid (c - d)$ $\Rightarrow m \mid (a - b) \cdot c$ und $m \mid (c - d) \cdot b$ $\Rightarrow m \mid (a - b) \cdot c + (c - d) \cdot b \Rightarrow \dots$
7.	$a \equiv b \pmod{m}$ $\Rightarrow a \cdot c \equiv b \cdot c \pmod{m}$	$6 \equiv 30 \pmod{12}$ $\Rightarrow 24 \equiv 120 \pmod{12}$	Wenn $a$ und $b$ bei Division durch $m$ den gleichen Rest lassen, dann auch $a \cdot c$ und $b \cdot c$ : siehe 6. mit $a \equiv b \pmod{m}$ und $c \equiv c \pmod{m}$
8.	<b>Potenzieren</b> $a \equiv b \pmod{m} \Rightarrow a^n \equiv b^n \pmod{m}$ <u>allgemein:</u> $n \in \mathbb{N}$ $a \equiv b \pmod{m} \Rightarrow a^n \equiv b^n \pmod{m}$	$3 \equiv 7 \pmod{4}$ $\Rightarrow 9 \equiv 49 \pmod{4}$ $\Rightarrow 27 \equiv 343 \pmod{4}$	Kongruenzen können potenziert werden: siehe 6. mit $a \equiv b \pmod{m}$ und $c \equiv c \pmod{m}$
9.	<b>Division</b> $\text{ggT}(d, m) = 1$ und $a \equiv b \pmod{m}$ $\Rightarrow \frac{a}{d} \equiv \frac{b}{d} \pmod{m}$	$\text{ggT}(3, 8) = 1$ und $6 \equiv 54 \pmod{8}$ $\Rightarrow 2 \equiv 18 \pmod{8}$  <b>Falls <math>\text{ggT}(d, m) \neq 1</math>:</b> $\text{ggT}(4, 8) = 4$ und $12 \equiv 28 \pmod{8}$ aber $3 \not\equiv 7 \pmod{8}$  Division kann aber auch funktionieren: $\text{ggT}(3, 6) = 3$ und $24 \equiv 6 \pmod{6}$ doch $8 \not\equiv 2 \pmod{6}$	Kongruenzen können durch eine ganze Zahl $d$ geteilt werden, wenn $d$ und der Modul teilerfremd sind ( $a$ und $b$ müssen durch $d$ teilbar sein): $d$ ist Teiler von $a$ und von $b$ : $a = a_1 d$ ; $b = b_1 d$ $a \equiv b \pmod{m} \Rightarrow m \mid (a - b)$ $\Rightarrow m \mid (a_1 d - b_1 d) \Rightarrow m \mid (a_1 - b_1) d$ $\text{ggT}(m, d) = 1 \Rightarrow m$ teilt nicht $d$ $\Rightarrow m \mid (a_1 - b_1) \Rightarrow a_1 \equiv b_1 \pmod{m}$ $\Rightarrow \frac{a}{d} \equiv \frac{b}{d} \pmod{m}$
10.	<b>Übergang zu den Resten</b> $a \equiv m \cdot c + r \pmod{m}$ $\Rightarrow a \equiv r \pmod{m}$	$80 \equiv 20 \pmod{6}$ $80 \equiv 6 \cdot 3 + 2 \pmod{6}$ $\Rightarrow 80 \equiv 2 \pmod{6}$	Bei der Rechnung mit Kongruenzen kann man immer zu den Resten übergehen. $a \equiv m \cdot c + r \pmod{m} \Rightarrow a - r \equiv m \cdot c \pmod{m}$ $m \mid m \cdot c \Rightarrow m \mid (a - r) \Rightarrow a \equiv r \pmod{m}$

## Faktorisierung großer Zahlen

- `a:=nextprime(10^14);`  
`b:=nextprime(a+1);`  
`m:=a*b;`

```

000000000000031
100000000000067
100000000000009800000000002077

```

- `Startzeit:=time();`  
`isprime(m);`  
`(time()-Startzeit)*ms;`

```

FALSE
67 ms

```

- `Startzeit:=time();`  
`ifactor(m);`  
`(time()-Startzeit)*ms;`

```

[1, 1000000000000031, 1, 1000000000000067, 1]
52129 ms

```

## Multiplikation zweier großer Primzahlen

- ⇒ Faktorisierung dauert lange  
(Ausnutzung beim RSA – Verschlüsselungsverfahren)
- ⇒ große Primzahlen für Geheimcodes
- ⇒ **SCHNELLERE PRIMZAHLTESTS**

## Kleiner Satz von Fermat

Ist  $a \in \mathbb{N}$  kein Vielfaches der Primzahl  $p$ , so ist  $a^{p-1} \equiv 1 \pmod{p}$ .

**Beispiel:  $p = 7$**

a	a Vielfaches von 7?	$a^6$	Rest von $a^6$ bei Division durch 7	$a^{7-1} \equiv 1 \pmod{7}$ ?
1	nein	$1^6 = 1$	$= 0 \cdot 7 + \underline{1}$	ja
2	nein	$2^6 = 64$	$= 9 \cdot 7 + \underline{1}$	ja
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				

Der kleine Satz von Fermat gilt nicht für zusammengesetzte Zahlen  $p$ , selbst dann nicht, wenn man die Voraussetzung „ $a$  kein Vielfaches von  $p$ “ in „ $\text{ggT}(a, p) = 1$ “ ändert.

**Beispiel:  $p = 8$**

a	a Vielfaches von 8?	$\text{ggT}(a, 8)$	$a^7$	Rest von $a^7$ bei Division durch 8	$a^{8-1} \equiv 1 \pmod{8}$ ?
1	nein	1	1	$= 0 \cdot 8 + \underline{1}$	ja
2					
3					
4					
5					
6					
7					
8					
9					
10					

Obwohl  $\text{ggT}(\_, 8) = \_$ , ist \_\_\_\_\_ modulo \_\_\_\_\_.

## Beweis des Kleinen Satzes von Fermat

**Kleiner Satz von Fermat:** Ist  $a \in \mathbb{N}$  kein Vielfaches der Primzahl  $p$ , so ist  $a^{p-1} \equiv 1 \pmod{p}$ .

Voraussetzung:  $a \in \mathbb{N}$  und  $\text{ggT}(a, p) = 1$  (a und p sind teilerfremd)

Behauptung:  $a^{p-1} \equiv 1 \pmod{p}$

Beweis:

Bei Division einer zu  $p$  teilerfremden Zahl durch  $p$  können nur die Reste  $1, 2, 3, 4, \dots, p-1$  auftreten.  
(Rest 0 würde bedeuten, dass die Zahl ein Vielfaches von  $p$  ist.)

**Menge aller von Null verschiedenen Reste bei Division durch  $p$ :**  $\{1, 2, 3, 4, \dots, p-1\}$

Alle diese  $p-1$  Reste sind **kleiner** als  $p$ . Außerdem ist  $p$  eine **Primzahl**.

$\Rightarrow$  Die Zahlen  $1, 2, 3, 4, \dots, p-1$  sind alle **teilerfremd zu  $p$** .

äquivalent:  $\text{ggT}(1, p) = 1; \text{ggT}(2, p) = 1; \dots; \text{ggT}(p-1, p) = 1$

Deswegen und da  $a$  kein Vielfaches von  $p$  ist, gilt mit  $r_i \in \{1, 2, 3, 4, \dots, p-1\}$  ( $i = 1, 2, \dots, p-1$ )

$1a$  kein Vielfaches von  $p \Rightarrow 1a$  lässt bei Division durch  $p$  einen Rest ungleich Null  $\Rightarrow 1a \equiv r_1 \pmod{p}$

$2a$  kein Vielfaches von  $p \Rightarrow 2a$  lässt bei Division durch  $p$  einen Rest ungleich Null  $\Rightarrow 2a \equiv r_2 \pmod{p}$

$3a$  kein Vielfaches von  $p \Rightarrow 3a$  lässt bei Division durch  $p$  einen Rest ungleich Null  $\Rightarrow 3a \equiv r_3 \pmod{p}$

...

$(p-1)a$  kein Vielfaches von  $p \Rightarrow (p-1)a$  lässt bei Division durch  $p$  einen Rest ungleich Null  $\Rightarrow (p-1)a \equiv r_{p-1} \pmod{p}$

Man kann nicht sagen, dass z.B. bei Division von  $3a$  durch  $p$  genau der Rest 3 entsteht, aber:

Bei Division der Zahlen  $a, 2a, 3a, \dots, (p-1)a$  durch  $p$  tritt jeder Rest  $1, 2, 3, 4, \dots, p-1$  genau einmal auf.  
(Auf der rechten Seite der Kongruenzen stehen alle verschiedene Zahlen.)

Angenommen zwei Reste der rechten Seite der Kongruenzen wären gleich:

$r_i = r_j$  mit  $i \neq j$   $i, j \in \{1, 2, 3, 4, \dots, p-1\}$

$ia \equiv r_i \pmod{p}$  und  $ja \equiv r_j \pmod{p} \Rightarrow ia \equiv ja \pmod{p}$  (siehe Kongruenzen 3. Rechenregel)

$\text{ggT}(a, p) = 1 \Rightarrow$  man kann durch  $a$  teilen  $\Rightarrow i \equiv j \pmod{p}$  (siehe Kongruenzen 9. Rechenregel)

$i$  und  $j$  sind **kleiner** als  $p \Rightarrow i = j \Rightarrow$  **Widerspruch**

$\Rightarrow$  Auf der rechten Seite der Kongruenzen tritt jeder Rest  $1, 2, 3, 4, \dots, p-1$  genau einmal auf.

Kongruenzen kann man multiplizieren:

(siehe Kongruenzen 6. Rechenregel)

$$1a \cdot 2a \cdot 3a \cdot \dots \cdot (p-1)a \equiv r_1 \cdot r_2 \cdot r_3 \cdot \dots \cdot r_{p-1} \pmod{p}$$

$$a^{p-1} \cdot 1 \cdot 2 \cdot 3 \cdot \dots \cdot (p-1) \equiv 1 \cdot 2 \cdot 3 \cdot \dots \cdot (p-1) \pmod{p} \quad (\text{jeder Rest genau einmal vorhanden})$$

Da die Reste  $1, 2, 3, 4, \dots, p-1$  alle zu  $p$  teilerfremd sind, kann man durch  $1, 2, 3, 4, \dots, p-1$  teilen:

(siehe Kongruenzen 9. Rechenregel)

$$\Rightarrow a^{p-1} \equiv 1 \pmod{p} \quad \text{q.e.d.}$$

## Umkehrung des Kleinen Satzes von Fermat

**Kleiner Satz von Fermat:** Ist  $a \in \mathbb{N}$  kein Vielfaches der Primzahl  $p$ , so ist  $a^{p-1} \equiv 1 \pmod{p}$ .

Die folgende Umkehrung des Kleinen Satzes von Fermat gilt nicht:<sup>A8</sup>

Gilt für alle Zahlen  $a \in \mathbb{N}$  mit  $\text{ggT}(a, n) = 1$ :  $a^{n-1} \equiv 1 \pmod{n}$ , dann ist  $n$  eine Primzahl.

**Bei einem möglichen Primzahltest reicht es, alle Zahlen  $a \in \mathbb{N}$  mit  $1 < a < n - 1$  zu überprüfen:**

- $a = 1$  muss nicht überprüft werden:  $1^{n-1} \equiv 1 \pmod{n}$
- $a = n - 1$  muss nicht überprüft werden:  $(n - 1)^{(n-1)} \equiv 1 \pmod{n}$ 

$$\begin{aligned} n - 1 &\equiv n - 1 && \pmod{n} \\ (n - 1)^2 &\equiv n^2 - 2n + 1 \equiv n \cdot (n - 2) + 1 \equiv 1 && \pmod{n} \\ (n - 1)^3 &\equiv 1 \cdot (n - 1) \equiv n - 1 && \pmod{n} \\ (n - 1)^4 &\equiv (n - 1)^2 \equiv 1 && \pmod{n} \\ &\dots && \\ (n - 1)^{\text{ungerade}} &\equiv n - 1 && \pmod{n} \\ (n - 1)^{\text{gerade}} &\equiv 1 && \pmod{n} \end{aligned}$$

$n$  gerade  $\Rightarrow \text{ggT}(2, n) = 2 \Rightarrow$  Test sollte schon eher entscheiden:  $n$  zusammengesetzt  
 $n$  ungerade ( $n - 1$  ist dann gerade):  $(n - 1)^{(n-1)} \equiv 1 \pmod{n}$

- $a = n$  muss nicht überprüft werden:  $\text{ggT}(n, n) = n$
- $a > n$  muss nicht überprüft werden: Es entstehen keine neuen Reste.

$a > n$  und  $\text{ggT}(a, n) = 1 \Rightarrow$  Es gibt Zahlen  $b, c \in \mathbb{N}$  und  $1 \leq c < n$ , so dass  $a = bn + c$

Es gilt:  $\text{ggT}(bn + c, n) = 1 \Rightarrow (bn + c)^{(n-1)} \equiv c^{n-1} \pmod{n}$

$$\begin{aligned} bn + c &\equiv c && \pmod{n} \\ (bn + c)^2 &\equiv b^2 n^2 + 2bnc + c^2 \equiv c^2 && \pmod{n} \\ (bn + c)^3 &\equiv c^3 && \pmod{n} \\ (bn + c)^4 &\equiv c^4 && \pmod{n} \\ &\dots && \\ (bn + c)^{n-1} &\equiv c^{n-1} && \pmod{n} \end{aligned}$$

Es gilt schon: Für alle  $a \in \mathbb{N}$  ( $1 < a < n - 1$ ) mit  $\text{ggT}(a, n) = 1$ :  $a^{n-1} \equiv 1 \pmod{n}$

$\Rightarrow$  Die Zahlen  $c \in \{1, 2, \dots, n - 1\}$  wurden schon getestet:  $\text{ggT}(c, n) = 1 \Rightarrow c^{n-1} \equiv 1 \pmod{n}$

$\Rightarrow$  Wenn schon für alle  $a < n$  alle Potenzen bei Division durch  $n$  den Rest 1 lassen, so auch alle Potenzen mit  $a > n$ .

<sup>A8</sup> Die Voraussetzung « $a$  kein Vielfaches von  $n$ » wurde zu « $\text{ggT}(a, n) = 1$ » verschärft.

## Carmichaelzahlen

Sei  $a \in \mathbb{N}$  und  $n \in \mathbb{N}$  zusammengesetzt.

Gilt für alle  $a$  mit  $\text{ggT}(a, n) = 1$ :  $a^{n-1} \equiv 1 \pmod{n}$ , so heißt  $n$  *Carmichaelzahl*.

### Carmichaelzahlen bis 300.000

561 = 3 · 11 · 17
1105 = 5 · 13 · 17
1729 = 7 · 13 · 19
2465 = 5 · 17 · 29
2821 = 7 · 13 · 31
6601 = 7 · 23 · 41
8911 = 7 · 19 · 67
10585 = 5 · 29 · 73
15841 = 7 · 31 · 73
29341 = 13 · 37 · 61
41041 = 7 · 11 · 13 · 41
46657 = 13 · 37 · 97
52633 = 7 · 73 · 103
62745 = 3 · 5 · 47 · 89
63973 = 7 · 13 · 19 · 37
75361 = 11 · 17 · 31
101101 = 7 · 11 · 13 · 101
115921 = 13 · 37 · 241
126217 = 7 · 13 · 19 · 73
162401 = 17 · 41 · 233
172081 = 7 · 13 · 31 · 61
188461 = 7 · 13 · 19 · 109
252601 = 41 · 61 · 101
278545 = 5 · 17 · 29 · 113
294409 = 37 · 73 · 109

### Wie viele Carmichaelzahlen gibt es?

- **25** Carmichaelzahlen bis 300.000      **25.997** Primzahlen bis 300.000  
 $\Rightarrow P(\text{Carmichaelzahl wird fälschlicherweise für Primzahl gehalten}) \approx 9,6 \cdot 10^{-4}$
- **2.163** Carmichaelzahlen bis  $25 \cdot 10^9$       **882.206.716** Primzahlen bis  $20 \cdot 10^9$   
 $\Rightarrow P(\text{Carmichaelzahl wird fälschlicherweise für Primzahl gehalten}) \approx 2,5 \cdot 10^{-6}$
- **105.212** Carmichaelzahlen bis  $10^{15}$       **29.844.570.422.669** Primzahlen bis  $10^{15}$   
 $\Rightarrow P(\text{Carmichaelzahl wird fälschlicherweise für Primzahl gehalten}) \approx 3,5 \cdot 10^{-9}$

## Pseudoprimezahlen

Sei  $a \in \mathbb{N}$  und  $n$  eine ungerade zusammengesetzte Zahl.

Gibt es ein  $a$  mit  $a^{n-1} \equiv 1 \pmod{n}$  so heißt  $n$  *Pseudoprimezahl zur Basis  $a$* .

### Pseudoprimezahlen bis 100.000

2	3	5	7	2, 3	2, 3, 5	2, 3, 5, 7
341	91	217	25	1105	1729	29341
561	121	561	325	1729	2821	46657
645	671	781	561	2465	6601	75361
1105	703	1541	703	2701	8911	
1387	949	1729	817	2821	15841	
1729	1105	1891	1105	6601	29341	
1905	1541	2821	1825	8911	41041	
2047	1729	4123	2101	10585	46657	
2465	1891	5461	2353	15841	52633	
2701	2465	5611	2465	18721	63973	
2821	2665	5731	3277	29341	75361	
3277	2701	6601	4525	31621		
4033	2821	7449	4825	41041		
4369	3281	7813	6697	46657		
4371	3367	8029	8321	49141		
4681	3751	8911	10225	52633		
5461	4961	9881	10585	63973		
6601	5551	11041	10621	75361		
7957	6601	12801	11041	83333		
8321	7381	13021	11521	83665		
8481	8401	13333	12025	8561		
8911	8911	13981	13665	90751		
10261	10585	14981	14089	93961		
10858	11011	15751	16725			
11305	12403	15841	18721			
12801	14383	16297	19345			
13741	15203	17767	20197			
13747	15457	21361	20417			
13981	15841	22791	20425			
14491	16471	23653	22945			
15709	16531	24211	25829			
15841	18721	25327	26419			
16705	19345	25351	29341			
18705	23521	29341	29857			
18721	24661	29539	29891			
19951	24727	30673	30025			
23001	28009	32021	30811			
23377	29161	35371	33227			
...	...	...	...			

Es gibt **9.592** Primzahlen bis **100.000**

### Pseudoprimezahlen bis 100.000

Basis	Anzahl	P(Fehlentscheidung)
2	78	0,0081
2, 3	23	0,0024
2, 3, 5	11	0,0011
2, 3, 5, 7	3	0,0003

Es gibt **882.206.716** Primzahlen bis  $20 \cdot 10^9$

### Pseudoprimezahlen bis $25 \cdot 10^9$

Basis	Anzahl	P(Fehlentscheidung)
2	21853	$2,5 \cdot 10^{-5}$
2, 3	4709	$5,3 \cdot 10^{-6}$
2, 3, 5	2552	$2,9 \cdot 10^{-6}$
2, 3, 5, 7	1770	$2,0 \cdot 10^{-6}$

## Übersicht über Primzahltests

### Primzahltest 1

Gilt für alle Zahlen  $a \in \mathbb{N}$  ( $1 < a < n$ ):  $\text{ggT}(a, n) = 1 \Rightarrow n$  ist Primzahl

- ```

isprime1:=proc(n)    local a;
begin
  if n < 2 then return(FALSE); end_if;
  for a from 2 to n-1 do
    if igcd(a, n) > 1 then return(FALSE); end_if;
  end_for;
  return(TRUE);
end_proc;

```

### Primzahltest 2

Gilt für alle Zahlen  $a \in \mathbb{N}$  ( $1 < a \leq \lfloor \sqrt{n} \rfloor$ ):  $\text{ggT}(a, n) = 1 \Rightarrow n$  ist Primzahl.

- ```

isprime2:=proc(n)    local a;
begin
  if n < 2 then return(FALSE); end_if;
  for a from 2 to trunc(sqrt(n)) do
    if igcd(a, n) > 1 then return(FALSE); end_if;
  end_for;
  return(TRUE);
end_proc;

```

### Kleiner Satz von Fermat

Ist  $a \in \mathbb{N}$  kein Vielfaches der Primzahl  $p$ , so ist  $a^{p-1} \equiv 1 \pmod{p}$ .

### Folgerung

Gibt es ein  $a \in \mathbb{N}$  mit  $\text{ggT}(a, n) = 1$  und  $a^{n-1} \not\equiv 1 \pmod{n} \Rightarrow n$  ist zusammengesetzt

### Primzahltest 3

Gilt für alle  $a \in \mathbb{N}$  ( $1 < a < n$ ) mit  $\text{ggT}(a, n) = 1$ :  $a^{n-1} \equiv 1 \pmod{n}$   
 $\Rightarrow n$  ist Primzahl oder Carmichaelzahl.

### Carmichaelzahlen

Sei  $n \in \mathbb{N}$  eine zusammengesetzte Zahl.

Gilt für alle  $a \in \mathbb{N}$  mit  $\text{ggT}(a, n) = 1$ :  $a^{n-1} \equiv 1 \pmod{n}$ , so heißt  $n$  Carmichaelzahl.

### Primzahltest 4

Gilt für einige  $a \in \mathbb{N}$  ( $1 < a < n - 1$ ):  $\text{ggT}(a, n) = 1$  und  $a^{n-1} \equiv 1 \pmod{n}$   
 $\Rightarrow n$  ist Primzahl oder Pseudoprimzahl.

### Pseudoprimzahlen

Sei  $n$  eine ungerade zusammengesetzte Zahl.

Gibt es ein  $a \in \mathbb{N}$  mit  $a^{n-1} \equiv 1 \pmod{n}$ , so heißt  $n$  Pseudoprimzahl zur Basis  $a$ .

## Carmichaelzahlen und Pseudoprimzahlen

### 1. Carmichaelzahlen

Erstellen Sie eine Prozedur *isCarmichael*, die eine Zahl testet, ob sie eine Carmichaelzahl ist.

**Definition:** Sei  $n \in \mathbb{N}$  eine zusammengesetzte Zahl.

Gilt für alle  $a \in \mathbb{N}$  mit  $\text{ggT}(a, n) = 1$ :  $a^{n-1} \equiv 1 \pmod{n}$ , so heißt  $n$  Carmichaelzahl.

**Hinweise:** Es reicht, alle Zahlen  $a \in \mathbb{N}$  mit  $1 < a < n$  und  $\text{ggT}(a, n) = 1$  zu testen.

Eingabe: positive Zahl  $n$

Ausgabe:  $n$  Carmichaelzahl  $\Rightarrow$  **TRUE**;  $n$  keine Carmichaelzahl  $\Rightarrow$  **FALSE**

**im Detail:** 1. Ist  $n$  eine Primzahl, dann ist  $n$  keine Carmichaelzahl. (Nutzen Sie *isprime(n)*.)

2. Teste alle Zahlen  $a$  von 2 bis  $n-1$ :

Wenn  $\text{ggT}(a, n) = 1$  und  $a^{n-1} \not\equiv 1 \pmod{n}$ , dann ist  $n$  keine Carmichaelzahl.

(Nutzen Sie **igcd(a, n)** und **a^(n-1) mod n**.)

3. Da nun alle Bedingungen der Definition erfüllt sind, ist  $n$  eine Carmichaelzahl.

**Fragen:** 1. Welche der Zahlen sind Carmichaelzahlen? 341, 401, 561, 1009, 1011, 1105, 1729, 2465

Die Zahlen \_\_\_\_\_ sind Carmichaelzahlen.

2. Was fällt Ihnen bei dem Test bezüglich der Rechenzeiten auf?  
Wann dauert der Test lange? Warum?

---



---

### 2. Pseudoprimzahlen zu einer vorgegebenen Basis $a$

Erstellen Sie eine Prozedur *isPseudoprime*, die eine Zahl testet, ob sie Pseudoprimzahl zur Basis  $a$  ist.

**Definition:** Sei  $n$  eine ungerade zusammengesetzte Zahl.

Gibt es ein  $a \in \mathbb{N}$  mit  $a^{n-1} \equiv 1 \pmod{n}$ , so heißt  $n$  Pseudoprimzahl zur Basis  $a$ .

**Hinweise:** Eingabe: positive Zahl  $n$  und positive Zahl  $a$

Ausgabe:  $n$  Pseudoprimzahl zur Basis  $a \Rightarrow$  **TRUE**;

$n$  keine Pseudoprimzahl zur Basis  $a \Rightarrow$  **FALSE**

**im Detail:** 1. Ist  $n$  gerade, dann ist  $n$  keine Pseudoprimzahl. (Nutzen Sie **igcd(?, n)**.)

2. Ist  $n$  eine Primzahl, dann ist  $n$  keine Pseudoprimzahl.

3. Ist  $a^{n-1} \equiv 1 \pmod{n}$ , dann ist  $n$  eine Pseudoprimzahl zur Basis  $a$ .

4. Da  $a^{n-1} \not\equiv 1 \pmod{n}$ , ist  $n$  keine Pseudoprimzahl zur Basis  $a$ .

**Fragen:** 1. Welche der folgenden Zahlen sind Pseudoprimzahlen zur Basis  $a$ ? (Angaben:  $(n, a)$ )  
(15, 4), (341, 2), (341, 3), (561, 3), (3608, 4), (5001, 6)

Die Zahlen \_\_\_\_\_  
sind Pseudoprimzahlen zur vorgegebenen Basis  $a$ .

### 3. Pseudoprimzahlen ohne vorgegebene Basis

Erstellen Sie eine Prozedur *isPseudoprime1*, die eine Zahl testet, ob sie eine Pseudoprimzahl ist.

**Hinweise:** Es reicht, alle Zahlen  $a \in \mathbb{N}$  mit  $1 < a < n-1$  zu testen.

Eingabe: positive Zahl  $n$

Ausgabe:  $n$  Pseudoprimzahl  $\Rightarrow$  **TRUE**,  $a$ ;  $n$  keine Pseudoprimzahl  $\Rightarrow$  **FALSE**

**im Detail:** Teste alle Zahlen  $a$  von 2 bis  $n-2$ : Wenn  $a^{n-1} \equiv 1 \pmod{n}$ , dann ist  $n$  eine Pseudoprimzahl.  
(Nutzen Sie zur Ausgabe: **return(TRUE, a)**.)

**Fragen:** 1. Welche der Zahlen sind Pseudoprimzahlen? Geben Sie in Klammern die jeweilige Basis an.  
15, 341, 561, 3608, 5001, 7004, 2465, 23001

Die Zahlen \_\_\_\_\_ sind Pseudoprimzahlen.

2. Welche Basis gibt der Test jeweils aus? \_\_\_\_\_

## Carmichaelzahlen und Pseudoprimzahlen

Seite 2

**Zusatz****1. Liste von Carmichaelzahlen**

Erstellen Sie eine Prozedur *CarmiachaelListe*, die alle Carmichaelzahlen innerhalb eines Intervalls in eine Liste schreibt.

**Hinweise:** Eingabe: Anfang und Ende des Intervalls  
Ausgabe: Liste mit allen Carmichaelzahlen von Anfang bis Ende

**im Detail:** Nutzen Sie die Prozedur *isCarmichael*

**Fragen:** 1. Schreiben Sie alle Carmichaelzahlen zwischen 1 und 2000 auf.

- 
2. Bei welchen Zahlen zwischen 1 und 1000 versagt der folgende Primzahltest?  
Primzahltest: Alle Zahlen kleiner als 2 sind keine Primzahlen.  
Gilt für alle  $a \in \mathbb{N}$  ( $1 < a < n - 1$ ) mit  $\text{ggT}(a, n) = 1$ :  $a^{n-1} \equiv 1 \pmod{n}$   
 $\Rightarrow n$  ist Primzahl
- 

**2. Liste von Pseudoprimzahlen zu einer vorgegebenen Basis**

Erstellen Sie eine Prozedur *PseudoprimeListe*, die alle Pseudoprimzahlen zu einer vorgegebenen Basis innerhalb eines Intervalls in eine Liste schreibt.

**Hinweise:** Eingabe: Basis  $a$ , Anfang und Ende des Intervalls  
Ausgabe: Liste mit allen Pseudoprimzahlen zur Basis  $a$  von Anfang bis Ende

**im Detail:** Nutzen Sie die Prozedur *isPseudoprime*

**Fragen:** 1. Schreiben Sie alle Pseudoprimzahlen zur Basis 6 zwischen 1000 und 2000 auf.

- 
2. Welche Zahlen zwischen 1 und 5000 sind Pseudoprimzahlen zu allen Basen 2, 3 und 5?
- 

3. Bei welchen Zahlen zwischen 1 und 5000 versagt der folgende Primzahltest?  
Primzahltest: Die Zahlen 2, 3 und 5 sind Primzahlen.  
Gilt für  $a = 2, 3, 5$ :  $\text{ggT}(a, n) = 1$  und  $a^{n-1} \equiv 1 \pmod{n}$   
 $\Rightarrow n$  ist Primzahl
-

## Primzahltests 3 und 4

1. Erstellen Sie eine Prozedur *isprime3*, die mit dem folgenden Primzahltest entscheidet, ob eine gegebene Zahl  $n$  Primzahl ist.

**Test 3:** Alle Zahlen kleiner als 2 sind keine Primzahlen.  
Gilt für alle  $a \in \mathbb{N}$  ( $1 < a < n$ ) mit  $\text{ggT}(a, n) = 1$ :  $a^{n-1} \equiv 1 \pmod{n} \Rightarrow n$  ist Primzahl

**Hinweise:** Eingabe: Zahl  $n$                       Ausgabe:  $n$  Primzahl  $\Rightarrow$  **TRUE**;  $n$  keine Primzahl  $\Rightarrow$  **FALSE**

**Fragen:** 1. Welche der Zahlen 10, 17, 307, 341, 561, 1105, 1729 werden als Primzahlen erkannt?

---

2. Bei welchen der unter 1. genannten Zahlen versagt der Test? Wie heißen die Zahlen?

---

2. Erstellen Sie eine Prozedur *isprime4*, die mit dem folgenden Primzahltest entscheidet, ob eine gegebene Zahl  $n$  Primzahl ist.

**Test 4:** Alle Zahlen kleiner als 2 sind keine Primzahlen.  
Die Zahlen 2 und 3 und 5 sind Primzahlen.  
Gilt für  $a = 2, 3$  und  $5$ :  $\text{ggT}(a, n) = 1$  und  $a^{n-1} \equiv 1 \pmod{n} \Rightarrow n$  ist Primzahl

**Fragen:** 1. Welche der Zahlen 10, 17, 307, 341, 561, 1105, 1729 werden als Primzahlen erkannt?

---

2. Bei welchen der unter 1. genannten Zahlen versagt der Test? Wie heißen die Zahlen?

---

3. Vergleichen Sie die Prozeduren *isprime*, *isprime1*, *isprime2*, *isprime3* und *isprime4* bezüglich der benötigten **Rechenzeit** bei Überprüfung der gegebenen Primzahlen und tragen Sie die Messwerte in die Tabelle ein. Nutzen Sie `startzeit:=time(): isprime?(?); (time()-startzeit)*ms`; **Messen Sie die Rechenzeit eines Tests bei den folgenden größeren Zahlen nicht mehr, sobald die Rechenzeit des Tests einmal 1000 ms überschritten hat.**

### Rechenzeiten der Primzahltests in Millisekunden

Primzahl	isprime	isprime1	isprime2	isprime3	isprime4	
1009						
10007						
100003						
100000000003						
$2^{600} + 187$						

Woran liegt es, dass die einzelnen Tests so viel Zeit benötigen?

*isprime1*: \_\_\_\_\_

*isprime2*: \_\_\_\_\_

*isprime3*: \_\_\_\_\_

*isprime4*: \_\_\_\_\_

Verändern Sie *isprime4* in *isprime4powermod*, indem Sie für die Berechnung der Reste statt z.B.  $2^{n-1} \bmod n$  den Befehl `powermod(2, n-1, n)` benutzen. Messen Sie die Rechenzeiten bei Überprüfung der Primzahlen für die neue Prozedur und tragen Sie die Messwerte in die Tabelle ein.

## Primzahltests 3 und 4

Seite 2

4. Vergleichen Sie die Prozeduren *isprime*, *isprime1*, *isprime2*, *isprime3* und *isprime4* bezüglich der **Sicherheit** ihrer Testaussagen. Geben Sie für jeden Primzahltest an, was das Ergebnis **TRUE** und was das Ergebnis **FALSE** bei Eingabe einer natürlichen Zahl größer als 1 jeweils bedeutet.

## Sicherheit der Primzahltests

Test	TRUE	FALSE
<i>isprime</i>		
<i>isprime1</i>		
<i>isprime2</i>		
<i>isprime3</i>		
<i>isprime4</i>		

## Zusatz

1. Erstellen Sie eine Prozedur *isprime4Liste*, die alle mit *isprime4powermod* erkannten Primzahlen in eine Liste schreibt.

**Hinweise:** Eingabe: Zahl  $n$  Ausgabe: Liste mit von *isprime4powermod* erkannten Primzahlen von 1 bis  $n$

- Fragen:**
- Wie viele Zahlen werden im Intervall von 1 bis 5000 als Primzahlen erkannt? \_\_\_\_\_
  - Welche der erkannten Zahlen von 1 bis 5000 sind keine Primzahlen?

- 
- Überprüfen Sie Ihre Aussage bei 2.: Erstellen Sie eine Primzahlliste von 1 bis 5000 mit dem *isprime*- Befehl von MuPAD. Vergleichen Sie die Anzahl der Primzahlen. Welche Zahlen wurden bei *isprime4powermod* fälschlicherweise für Primzahlen gehalten?
- 

2. Erstellen Sie eine Prozedur *isprime5*, die mit dem folgenden Primzahltest entscheidet, ob eine gegebene Zahl  $n$  Primzahl ist.

**Test 5:** Alle Zahlen kleiner als 2 sind keine Primzahlen.  
Die Zahlen 2, 3 und 5 sind Primzahlen.

Gilt für  $a = 2, 3, 5$ :  $\text{ggT}(a, n) = 1$  und  $a^{\frac{n-1}{2}} \equiv \pm 1 \pmod{n} \Rightarrow n$  ist Primzahl

**Hinweise:** Der Operator *powermod* von MuPAD berechnet jeweils die positiven Reste.

- Fragen:**
- Welche der Zahlen 10, 17, 307, 341, 561, 1105, 1729, 2821 werden als Primzahlen erkannt?
- 

- Bei welchen der unter 1. aufgeführten Zahlen versagt der Test?
- 

- Vergleichen Sie *isprime5* bezüglich der Sicherheit und der Rechenzeit mit den anderen Primzahltests. Was ist der Vorteil dieses Tests gegenüber *isprime4*?
- 

- Erstellen Sie mit *isprime5* eine Prozedur *isprime5Liste*. Welche Zahlen bestimmt *isprime5* auf den ersten 15000 Zahlen falsch? Wie viele Zahlen bestimmt *isprime5* auf den ersten 20000 (50000) Zahlen falsch?
-

## Carmichaelzahlen und Pseudoprimzahlen in MuPAD

- `isCarmichael:=proc(n) local a;`  
`begin`  
`if n<2 then return(FALSE); end_if;`  
`if isprime(n) = TRUE then return(FALSE); end_if;`  
`for a from 2 to n - 1 do`  
`if igcd(a, n) = 1 and a^(n-1) mod n <> 1 then return(FALSE); end_if`  
`end_for;`  
`return(TRUE);`  
`end_proc:`
- `isPseudoprime:=proc(n,a)`  
`begin`  
`if igcd(2, n) = 2 then return(FALSE); end_if;`  
`if isprime(n) = TRUE then return(FALSE); end_if;`  
`if a^(n-1) mod n = 1 then return(TRUE); end_if;`  
`return(FALSE);`  
`end_proc:`
- `isPseudoprime1:=proc(n) local a;`  
`begin`  
`if igcd(2, n) = 2 then return(FALSE); end_if;`  
`if isprime(n) = TRUE then return(FALSE); end_if;`  
`for a from 2 to n - 2 do`  
`if a^(n-1) mod n = 1 then return(TRUE, a); end_if`  
`end_for;`  
`return(FALSE);`  
`end_proc:`
- `CarmichaelListe:=proc(Anfang, Ende) local i, Liste;`  
`begin`  
`Liste:=[];`  
`for i from Anfang to Ende do`  
`if isCarmichael(i) = TRUE then Liste:=append(Liste, i); end_if;`  
`end_for;`  
`return(Liste);`  
`end_proc:`
- `PseudoprimeListe:=proc(a,Anfang,Ende) local i, Liste;`  
`begin`  
`Liste:=[];`  
`for i from Anfang to Ende do`  
`if isPseudoprime(i,a) = TRUE then Liste:=append(Liste, i); end_if;`  
`end_for;`  
`return(Liste);`  
`end_proc:`
- `psp2:=PseudoprimeListe(2,1,5000);`  
`psp3:=PseudoprimeListe(3,1,5000);`  
`psp5:=PseudoprimeListe(5,1,5000);`
- `Menge_psp2:={op(psp2)};`  
`Menge_psp3:={op(psp3)};`  
`Menge_psp5:={op(psp5)};`
- `Menge_psp2 intersect Menge_psp3 intersect Menge_psp5;`

## Primzahltests in MuPAD

- `isprime1:=proc(n) local a;`  
`begin`  
`if n < 2 then return(FALSE); end_if;`  
`for a from 2 to n-1 do`  
`if igcd(a,n) > 1 then return(FALSE); end_if;`  
`end_for;`  
`return(TRUE);`  
`end_proc:`
- `isprime2:=proc(n) local a;`  
`begin`  
`if n < 2 then return(FALSE); end_if;`  
`for a from 2 to trunc(sqrt(n)) do`  
`if igcd(a,n) > 1 then return(FALSE); end_if;`  
`end_for;`  
`return(TRUE);`  
`end_proc:`
- `isprime3:=proc(n) local a;`  
`begin`  
`if n<2 then return(FALSE); end_if;`  
`for a from 2 to n-1 do`  
`if igcd(a, n) = 1 and a^(n-1) mod n <> 1 then return(FALSE); end_if;`  
`end_for;`  
`return(TRUE);`  
`end_proc:`
- `isprime4:=proc(n)`  
`begin`  
`if n<2 then return(FALSE); end_if;`  
`if n=2 or n=3 or n=5 then return(TRUE); end_if;`  
`if igcd(2, n) <> 1 or 2^(n-1) mod n <> 1 then return(FALSE); end_if;`  
`if igcd(3, n) <> 1 or 3^(n-1) mod n <> 1 then return(FALSE); end_if;`  
`if igcd(5, n) <> 1 or 5^(n-1) mod n <> 1 then return(FALSE); end_if;`  
`return(TRUE);`  
`end_proc:`
- `isprime4powermod:=proc(n)`  
`begin`  
`if n<2 then return(FALSE); end_if;`  
`if n=2 or n=3 or n=5 then return(TRUE); end_if;`  
`if igcd(2, n) <> 1 or powermod(2,n-1,n) <> 1 then return(FALSE); end_if;`  
`if igcd(3, n) <> 1 or powermod(3,n-1,n) <> 1 then return(FALSE); end_if;`  
`if igcd(5, n) <> 1 or powermod(5,n-1,n) <> 1 then return(FALSE); end_if;`  
`return(TRUE);`  
`end_proc:`
- `isprime?Liste:=proc(n) local i, Liste;`  
`begin`  
`Liste:=[];`  
`for i from 1 to n do`  
`if isprime?(i)=TRUE then Liste:=append(Liste, i); end_if;`  
`end_for;`  
`return(Liste, nops(Liste));`  
`end_proc:`
- `isprime5:=proc(n)`  
`begin`  
`if n<2 then return(FALSE); end_if;`  
`if n=2 or n=3 or n=5 then return(TRUE); end_if;`  
`if igcd(2, n) <> 1 or igcd(3, n) <> 1 or igcd(5, n) <> 1 then return(FALSE); end_if;`  
`if powermod(2,(n-1)/2,n) <> 1 and powermod(2,(n-1)/2,n) <> n-1 then return(FALSE); end_if;`  
`if powermod(3,(n-1)/2,n) <> 1 and powermod(3,(n-1)/2,n) <> n-1 then return(FALSE); end_if;`  
`if powermod(5,(n-1)/2,n) <> 1 and powermod(5,(n-1)/2,n) <> n-1 then return(FALSE); end_if;`  
`return(TRUE);`  
`end_proc:`

## Neue Verlaufsplanung

Die angesprochenen Besserungen der Arbeitsblätter und der Kontrolle sind zu berücksichtigen.

### Veränderte Übersicht über die Stundeneinteilung

Stunde	Inhalt	Bemerkungen zur Durchführung
1./2.	Zahlentheoretische Grundlagen	EA/UG: Wiederholung bekannter Begriffe; Diskussion von Teilbarkeitseigenschaften; Zusammenfassung für die Lernenden
	Bedeutung der Primzahlen	UG: Warum beschäftigte und beschäftigt man sich mit Primzahlen?
	<i>Unendlichkeit der Primzahlmenge</i>	LV/UG: Verteilung der <i>Primzahlen</i> ; Vermutungen der Lernenden LV: Satz über die <i>Unendlichkeit der Primzahlmenge</i> (ohne Beweis)
	Erste Primzahltests	UG: Aus der Primzahldefinition folgt Test 1. UG: Muss man alle Zahlen $a \in \mathbb{N}$ mit $1 < a < n$ testen? $\Rightarrow$ Test 2 LV: Beweis von Test 2
	Suche nach einer Primzahlfolge	UG: Ungerade Zahlen EA/UG: Polynome, die Primzahlen liefern LV: Definition der <i>Fermatschen</i> und <i>Mersenneschen Zahlen</i> HA: Überprüfung der <i>Fermatschen</i> und <i>Mersenneschen Zahlen</i> auf die Primzahleigenschaft
3.	Große Primzahlen und ihre Entdeckungen	UG: Vergleich der Hausaufgabe LV: <i>Fermatsche</i> und <i>Mersennesche Zahlen</i> ; Primzahlsuche im Internet
	Erste Primzahltests in MuPAD	PA/UG: Test 1 in MuPAD; Diskussion von Test 1 HA: Test 2 in MuPAD; Wiederholung: Potenzgesetze
4./5.	<i>Sieb des Eratosthenes</i> und <i>Sieb des Ulam</i>	EA: Arbeitsblatt UG: Vergleich der Siebe; Folgerungen aus dem <i>Sieb des Ulam</i>
	Kongruenzen; Kongruenzrechnung	UG: Gemeinsamkeiten der Zahlen in einer Spalte beim <i>Sieb des Ulam</i> LV: Kongruenzbegriff EA/UG: Ausgewählte Rechenregeln für Kongruenzen anhand von Beispielen; Zusammenfassung für die Lernenden
6.	Faktorisierung großer Zahlen	LV: Warum benötigt man große Primzahlen?
	<i>Kleiner Satz von Fermat</i> und sein Beweis	LV: <i>Kleiner Satz von Fermat</i> EA/UG: Erarbeitung des Satzes LV: Beweis des Satzes (Einsatz von MuPAD)
7./8.	Kongruenzrechnung	LV, EA: Reste berechnen UG: Vorteile des <i>Kleinen Satzes von Fermat</i> HA: Reste berechnen
	Umkehrung des <i>Kleinen Satzes von Fermat</i>	UG: Umkehrung des Satzes LV: Vorstellung der Tests 3 und 4; Definition der <i>Carmichaelzahlen</i> und <i>Pseudoprimzahlen</i> ; EA/UG: Güte der Tests und praktische Tauglichkeit
9.	<i>Carmichaelzahlen</i> und <i>Pseudoprimzahlen</i> in MuPAD	LV: <i>Carmichaelzahlen</i> , <i>Pseudoprimzahlen</i> und Primzahltests PA: Arbeitsblatt
10./11.	Primzahltests in MuPAD	PA: Arbeitsblatt UG: Primzahltests
12.	Kontrolle	EA: Aufgabenblatt