## Aufgabe 1 (Komplexitätsanalyse – Sieb des Eratosthenes) *30 Punkte*

Zur Ermittlung aller Primzahlen unter den natürlichen Zahlen bis zu einer maximalen Zahl $N \in \mathbb{N}$ kann man das *Sieb des Eratosthenes* verwenden.

Zunächst werden alle Zahlen 2, 3, ..., $N$ notiert. Alle Zahlen sind Primzahlkandidaten, bis sie als nicht prim markiert werden. Die kleinste unmarkierte Zahl, 2, ist eine Primzahl. Nun werden der Reihe nach alle Vielfachen dieser Primzahl unter den Primzahlkandidaten als zusammengesetzte Zahlen markiert. Anschließend erkennt man die kleinste verbleibende unmarkierte Zahl als Primzahl und markiert ihre Vielfache usw. Nach Abschluss des Verfahrens bilden alle unmarkierten Zahlen die Menge der Primzahlen von 2 bis $N$.

Es ergibt sich folgender Algorithmus:

```
Algorithmus Eratosthenes(N : integer) A : array[2..N] of integer;
var i, j : integer;
var A : array[2..N] of integer;
begin
  for i := 2 to N do
    A[i] := 0;
  end for;
  j := 2;
  while j ≤ N do
    if A[j] = 0 then
      i := 2 * j;
      while i ≤ N do
        if A[i] = 0 then
          A[i] := j;
        end if;
        i := i + j;
      end while;
    end if
    j := j + 1;
  end while;
  return A;
end Eratosthenes.
```

Nach Ablauf des Algorithmus gilt für $2 \leq n \leq N$: $A[n] = 0$ gdw. $n$ ist prim.

Es sei $M(N) = M_{write}(N) + M_{read}(N)$ die Funktion, die die Laufzeitkosten von *Eratosthenes* in Abhängigkeit von dessen Argument $N$ berechnet. Dabei sollen ausschließlich Arrayzugriffe auf $A$ als Kosten zählen. Wir verwenden zwei Teilkostenfunktionen:

$M_{read}(N)$ zählt jeden *lesenden* Zugriff auf das Array mit Kostenmaß 1,
$M_{write}(N)$ zählt jeden *schreibenden* Zugriff auf das Array mit Kostenmaß 1,
die Kosten sonstiger Operationen sollen vernachlässigt werden.

*10 Punkte* (a) Bestimmen Sie die Teilkostenfunktion $M_{write}(N)$ und erläutern Sie deren Herleitung.

*15 Punkte* (b) Bestimmen Sie die Teilkostenfunktion $M_{read}(N)$ und erläutern Sie deren Herleitung.

*2 Punkte* (c) Bestimmen Sie die Gesamtkostenfunktion $M(N)$ von *Eratosthenes*.

*3 Punkte* (d) Geben Sie die Komplexitätsklasse von $M(N)$ in der O-Notation an.

**Hinweise**: Neben den Formeln im Beiheft dürften die folgenden Gleichungen und Abschätzungen hilfreich sein:

1. $\sum_{k=1}^{n} \frac{1}{k \ln k} \approx \ln \ln n$
2. Für $\pi_n$, die Anzahl der Primzahlen $\leq n$, gilt: $\pi_n \cong \frac{n}{\ln n}$.
3. Für die $n$-te Primzahl gilt: $p_n \cong n \ln n$.
4. $\ln\left(\frac{n}{\ln n}\right) = \ln n - \ln \ln n \cong \ln n$

**Anmerkung**: Eigentlich genügt es auch, wenn man mit dem Streichen der Vielfachen von $j$ bei $j^2$ beginnt, da alle kleineren Vielfachen bereits markiert sind. Auch kann man den Algorithmus abbrechen, sobald $j^2$ größer als die Schranke $N$ ist. Zur Vereinfachung der Analyse wollen wir hier jedoch freigiebiger sein.

***25 Punkte***

## Aufgabe 2 (Auto-Algebra)

Das Ziel dieser Aufgabe ist es, ein Auto mit Hilfe einer Algebra zu modellieren. Dabei wird der Zustand eines Autos durch die folgenden Angaben beschrieben:

- Höchstgeschwindigkeit $h$ in km/h
- Gesamtlaufleistung $m$ des Motors in km
- Automatikgetriebe $a$ ja/nein
- Fahrbereitschaft $ok$ ja/nein

Die Algebra soll die folgenden Operationen enthalten:

*create* erzeugt ein fabrikneues Auto aus den Parametern Höchstgeschwindigkeit und Automatikgetriebe, wobei ungültige Werte sinnvoll auf die Ober- bzw. Untergrenze korrigiert werden;

*tune* durch Umbaumaßnahmen wird einem Auto eine neue Höchstgeschwindigkeit übergeben, die nicht kleiner und maximal ein Drittel größer als die bisherige Höchstgeschwindigkeit sein darf, wobei eine ungültige Angabe ignoriert wird;

*drive* ein Auto wird für einen bestimmten Zeitraum (in Stunden) mit einer bestimmten konstanten Geschwindigkeit in einem bestimmten Gang gefahren, wobei ein Automatikgetriebe bei Bedarf automatisch von einem angegebenen Vorwärtsgang in einen anderen Vorwärtsgang schaltet; bei ungültigen Eingaben wird das Auto nicht fortbewegt;

*repair* versetzt ein Auto in einen fahrbereiten Zustand;

*engine* ein Auto erhält einen anderen Motor mit einer bestimmten Höchstgeschwindigkeit und einer bestimmten Laufleistung und ist anschließend fahrbereit, wobei ungültige Eingaben das Auto unverändert lassen;

*compare* vergleicht die Fahrbereitschaft und die Höchstgeschwindigkeit (in dieser Reihenfolge) eines Autos mit denen eines anderen und gibt entweder 1 (bei Überlegenheit des ersten Autos), 0 (bei Gleichheit) oder $-1$ zurück;

*crash* simuliert einen Zusammenprall zweier Autos, die der Funktion zusammen mit ihrer jeweils aktuellen Geschwindigkeit übergeben werden, und gibt beide Autos zurück, dabei verlieren diese die Fahrbereitschaft, falls die Summe der Geschwindigkeiten 40 km/h übersteigt; bei ungültigen Geschwindigkeiten (über der Höchstgeschwindigkeit oder negativ) bleiben die Autos unverändert.

Es sei vorausgesetzt, dass jedes Auto genau sieben „Gänge“ hat, und zwar einen Rückwärtsgang $-1$, einen Leerlauf 0 und die Vorwärtsgänge 1 bis 5. Die Höchstgeschwindigkeit darf nicht kleiner als 80 und nicht größer als 400 km/h sein. Beachten Sie, dass für die Ausführung einer Operation evtl. zunächst Bedingungen überprüft werden müssen. Die Gesamtlaufleistung wird durch Fahren im Rückwärtsgang nicht verändert, im Leerlauf kann nicht gefahren werden. Es ist nicht möglich, mit einer höheren Geschwindigkeit als der Höchstgeschwindigkeit des Autos zu fahren. Ein Auto mit manuellem Getriebe ist nicht mehr fahrbereit und kommt sofort zum Stillstand, falls im ersten Gang mit über 40% oder im zweiten Gang mit über 80% der Höchstgeschwindigkeit gefahren wird. Unabhängig von der Getriebeart verliert ein Auto die Fahrbereitschaft bei einer Fahrt im Rückwärtsgang mit über 50 km/h.

(a) Geben Sie eine Signatur für die Algebra *car* an. *8 Punkte*

(b) Spezifizieren Sie geeignete Trägermengen, und geben Sie Funktionen für die Operationen an. *17 Punkte*

## Aufgabe 3 (O-Notation) *25 Punkte*

Vereinfachen Sie die folgenden Funktionen mit Hilfe der O-Notation:

(a) $T_1(n) = 5n - 7$

(b) $T_2(n) = 2\,n^3 + 16$

(c) $T_3(n) = k\,n^2 + 5$

(d) $T_4(n) = l\,n \log n + j$

(e) $T_5(n) = T_2(n) - T_1(n)$

(f) $T_6(n) = T_2(n) + T_3(n)$

(g) $T_7(n) = T_1(n) + T_4(n)$

(h) $T_8(n) = T_3(n) + T_4(n)$

(i) $T_9(n) = T_1(n) + T_2(n) + T_4(n)$

(j) $T_{10}(n) = T_1(n) \cdot T_2(n)$

(k) $T_{11}(n) = T_2(n) \cdot T_4(n)$

(l) $T_{12}(n) = (T_3(n))^2$

(m) $T_{13}(n) = (T_4(n))^2$

***20 Punkte*** **Aufgabe 4 (O-Notation)**

*5 Punkte* (a) Warum sagt die Behauptung „Die Laufzeit von Algorithmus $A$ ist mindestens $O(n^2)$“ nichts aus?

(b) Beweisen oder widerlegen Sie:

*5 Punkte* (i) $(n+1)! \stackrel{?}{=} O(n!)$

*5 Punkte* (ii) $(n+1)^n \stackrel{?}{=} O(n^n)$

Hinweis: Die Folge $\left(1 + \frac{1}{n}\right)^n$ ist streng monoton wachsend und hat den Grenzwert $e \approx 2,718$.

*5 Punkte* (iii) $n^{n+1} \stackrel{?}{=} O(n^n)$