Aufgabe 1

(a) Bestimmung der Laufzeitfunktion $M_{write}(N)$:

Die Initialisierung des Arrays (Zeilen 5-7) benötigt einmalig (N-1) Schreibzugriffe. Darüberhinaus wird offensichtlich für jede zusammengesetzte Zahl genau einmal die Markierung überschrieben. Dies sind insgesamt

$$N - \pi_N \overset{(2)}{=} N - \frac{N}{\ln N}$$

Schreibzugriffe. Arrayzellen für Primzahlen werden dagegen nach der Initialisierung nicht mehr beschrieben. Somit erhalten wir

$$M_{write}(N) = \underbrace{N-1}_{\text{Initialisierung}} + \underbrace{N - 1 - \frac{N}{\ln N}}_{\substack{\text{Markierung} \\ \text{zusammengesetzter} \\ \text{Zahlen}}}$$

$$= 2N - \frac{N}{\ln N} - 2$$

(b) Bestimmung der Laufzeitfunktion $M_{read}(N)$:

In der äußeren while-Schleife erfolgen in Zeile 10 insgesamt (N - 1) Lesezugriffe auf A.

Ist j eine Primzahl, so erfolgt in der inneren **while**-Schleife (Zeilen 12-17) die Markierung der bislang unmarkierten Vielfachen i von j, $j \le i \le N$. Dazu wird für jedes Vielfache i von j einmal aus A gelesen. Die Anzahl der Lesezugriffe auf A hängt vom jeweiligen j ab.

Da für eine Primzahl j in der inneren Schleife nur jedes j-te Element von A betrachtet wird, müssen (in Zeile 13) für die Primzahl j gerade N / j Arrayzellen gelesen werden. Abschätzung (3) erlaubt uns zudem, jede der π_N Primzahlen annähernd zu bestimmen. Daher ergibt sich folgende Anzahl an Lesevorgängen in der inneren Schleife:

$$\sum_{\substack{j=2, \\ j\,\text{prim}}}^{N} \left(\frac{N}{j} - 1 \right) = \sum_{k=1}^{\pi_N} \left(\frac{N}{p_k} - 1 \right) = \sum_{k=2}^{\pi_N} \left(\frac{N}{p_k} - 1 \right) + \frac{N}{2} - 1$$

$$\overset{(3)}{\approx} \sum_{k=2}^{\pi_N} \left(\frac{N}{k \ln k} - 1 \right) + \frac{N}{2} - 1 = N \cdot \sum_{k=2}^{\pi_N} \frac{1}{k \ln k} - (\pi_N - 1) + \frac{N}{2} - 1$$

$$\overset{(1)}{\approx} N \cdot \ln \ln \pi_N - \pi_N + \frac{N}{2} \overset{(2)}{\approx} N \cdot \ln\left(\ln \frac{N}{\ln N} \right) - \frac{N}{\ln N} + \frac{N}{2}$$

$$\overset{(4)}{\approx} N \cdot \ln \ln N - \frac{N}{\ln N} + \frac{N}{2} = N \left(\ln \ln N - \frac{1}{\ln N} + \frac{1}{2} \right)$$

Insgesamt ergibt sich für die Anzahl der Lesezugriffe:

$$M_{read}(N) = \underbrace{N\left(\ln\ln N - \frac{1}{\ln N} + \frac{1}{2}\right)}_{\text{Lesezugriffe innere Schleife}} + \underbrace{N-1}_{\substack{\text{Lesezugriffe}\\ \text{äußere Schleife}}}$$

$$= N\left(\ln\ln N - \frac{1}{\ln N} + \frac{3}{2}\right) - 1$$

(c) Bestimmung der Laufzeitfunktion $M(N)$:
Die Gesamtlaufzeit ergibt sich als Summe der Lese- und Schreibzugriffe auf A:

$$M(N) = \underbrace{N\left(\ln\ln N - \frac{1}{\ln N} + \frac{3}{2}\right) - 1}_{M_{read}(N)} + \underbrace{2N - \frac{N}{\ln N} - 2}_{M_{write}(N)}$$

$$= N\left(\ln\ln N - \frac{2}{\ln N} + \frac{7}{2}\right) - 3$$

(d) Bestimmung der Komplexitätsklasse in O-Notation:
Es gilt:

$$M(N) = O(N \ln\ln N) = O(N \log\log N)$$

Aufgabe 2

(a)

algebra *car*

sorts *car, gear, pos_real, comp, bool*

ops	*create*:	$pos\_real \times bool$	$\to$	car
	tune:	$car \times pos\_real$	$\to$	car
	drive:	$car \times pos\_real \times pos\_real \times gear$	$\to$	car
	repair:	car	$\to$	car
	engine:	$car \times pos\_real \times pos\_real$	$\to$	car
	compare:	$car \times car$	$\to$	$comp$
	crash:	$car \times pos\_real \times car \times pos\_real$	$\to$	$car \times car$

(b)

sets $bool = \{true, false\}$

$comp = \{-1, 0, 1\}$

$pos\_real = \{x \in \mathbb{R} \mid x \geq 0\}$

$gear = \{-1, 0, \ldots, 5\}$

$car = \{(h, m, a, ok) \mid h, m \in pos\_real, a, ok \in bool, 80 \leq h \leq 400\}$

functions

$$create(h, a) = \begin{cases} (h, 0, a, true) & \text{falls} \quad 80 \leq h \leq 400 \\ (400, 0, a, true) & \text{falls} \quad h > 400 \\ (80, 0, a, true) & \text{sonst} \end{cases}$$

$$tune((h, m, a, ok), h') = \begin{cases} (h', m, a, ok) & \text{falls } h < h' \leq \frac{4}{3} h \wedge h' \leq 400 \\ (h, m, a, ok) & \text{sonst} \end{cases}$$

$$drive((h, m, a, ok), t, v, g) = \begin{cases} (h, m + vt, a, ok) & \text{falls} \quad ok \wedge 0 \leq v \leq h \wedge g > 0 \wedge \\ & \quad (a \vee \neg((g = 1 \wedge v > 0.4h) \\ & \quad \wedge \neg(g = 2 \wedge v > 0.8h))) \\ (h, m, a, false) & \text{falls} \quad \neg ok \vee (((g = 1 \wedge v > 0.4h) \\ & \quad \vee (g = 2 \wedge v > 0.8h)) \wedge \neg a) \\ & \quad \vee (g = -1 \wedge v > 50) \\ (h, m, a, ok) & \text{sonst} \end{cases}$$

$$repair((h, m, a, ok)) = (h, m, a, true)$$

$$engine((h, m, a, ok), h', m') = \begin{cases} (h', m', a, true) & \text{falls} \quad 80 \leq h' \leq 400 \\ (h, m, a, ok) & \text{sonst} \end{cases}$$

$$compare((h, m, a, ok), (h', m', a', ok')) = \begin{cases} 1 & \text{falls} \quad (ok \wedge \neg ok') \vee (ok = ok' \wedge h > h') \\ 0 & \text{falls} \quad ok = ok' \wedge h = h' \\ -1 & \text{sonst} \end{cases}$$

$$crash((h, m, a, ok), v, (h', m', a', ok'), v') = \begin{cases} ((h, m, a, false), & \text{falls} \quad v + v' > 40 \\ (h', m', a', false)) & \quad \wedge 0 \leq v \leq h \wedge 0 \leq v' \leq h \\ ((h, m, a, ok), & \text{sonst} \\ (h', m', a', ok')) & \end{cases}$$

Aufgabe 3

(a) $T_1(n) = 5\,n - 7 = O(n)$

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

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

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

(e) $T_5(n) = T_2(n) - T_1(n) = O(n^3)$

(f) $T_6(n) = T_2(n) + T_3(n) = O(n^3)$

(g) $T_7(n) = T_1(n) + T_4(n) = O(n\ log\ n)$

(h) $T_8(n) = T_3(n) + T_4(n) = O(n^2)$

(i) $T_9(n) = T_1(n) + T_2(n) + T_4(n) = O(n^3)$

(j) $T_{10}(n) = T_1(n) \cdot T_2(n) = O(n^4)$

(k) $T_{11}(n) = T_2(n) \cdot T_4(n) = O(n^4 \log n)$

(l) $T_{12}(n) = (T_3(n))^2 = O(n^4)$

(m) $T_{13}(n) = (T_4(n))^2 = O((n \log n)^2)$ oder $O(n^2(\log n)^2)$; möglich ist auch die Schreibweise $O(n^2 \log^2 n)$.

Aufgabe 4

(a)

$O(n^2)$ enthält alle Funktionen, die höchstens so schnell wachsen wie $f(n) = n^2$. Die Aufgabenstellung besagt nun, dass die Laufzeit von A in $O(n^2)$ liegt oder schneller wächst als $f(n) = n^2$. Also erfüllen alle möglichen Laufzeitfunktionen die Aussage.

(b)

Entsprechend der Definition der O-Notation prüfen wir zur Beantwortung der Frage

$$f(n) \overset{?}{=} O(g(n))$$

jeweils, ob Konstanten c und n_0 existieren, für die gilt:

$$f(n) \leq c \cdot g(n) \quad \forall n \geq n_0$$

(i)

$(n+1)! \leq c \cdot n! \Leftrightarrow$

$(n+1) \cdot n! \leq c \cdot n! \Leftrightarrow$

$c \geq n+1$

Der Wert von c muß also immer größer sein als n, um die Ungleichung zu erfüllen. Deshalb ist es nicht möglich, eine entsprechende Konstante c anzugeben: $(n+1)! \notin O(n!)$.

(ii)

$(n+1)^n \le c \cdot n^n \Leftrightarrow$

$\left(\frac{n+1}{n}\right)^n \le c \Leftrightarrow$

$\left(1+\frac{1}{n}\right)^n \le c \Leftrightarrow$

$c \ge e, n_0 = 1$

Da $\left(1+\frac{1}{n}\right)^n$ streng monoton steigend gegen den Grenzwert e strebt, ist e eine Obergrenze für c für beliebige $n \ge 1$. Also gilt: $(n+1)^n = O(n^n)$.

(iii)

$n^{n+1} \le c \cdot n^n \Leftrightarrow$

$n \cdot n^n \le c \cdot n^n \Leftrightarrow$

$c \ge n$

Analog zu (i) können wir auch hier keine solche Konstante c angeben: $n^{n+1} \notin O(n^n)$.