Grundbegriffe der Informatik — Aufgabenblatt 7
Lösungsvorschläge

Matr.nr.:	
Nachname:	
Vorname:	
Tutorium Nr.:	Tutor*in:

Ausgabe: 03. Dezember 2020, 12:00 Uhr

Abgabe: 10. Dezember 2021, 12:30 Uhr
in dem Holzkasten neben Raum -119
im UG des Info-Gebäudes (50.34)

Lösungen werden nur korrigiert, wenn sie
- handschriftlich erstellt sind (Tablet-Ausdruck erlaubt) und
- mit dieser Seite als Deckblatt
- in der oberen **linken** Ecke zusammengeheftet

rechtzeitig abgegeben werden.

Abgaberegeln für Teilnehmer der Online-Tutorien:
- handschriftlich erstellt (lesbare Fotos akzeptiert)
- **rechtzeitig**, mit diesem Deckblatt in **genau einer** PDF-Datei
- direkt an den entsprechenden Tutor abgeben.

---

*Von Tutor*in auszufüllen:*

erreichte Punkte

Blatt 7:	/ 20 [+3]
Blätter 7 – 7:	/ 20 [+3]

Hinweis:
Ab diesem Aufgabenblatt sind alle Lösungen einzeln und nicht mehr in Zweiergruppen abzugeben.

Aufgabe 7.1 (0,5 + 1 + 2,5 = 4 Punkte)
Rechts ist ein Speicher $m : \mathbb{K}_6 \to \mathbb{K}_6$ tabellarisch dargestellt. Dabei sind Adressen und Werte im Zweierkomplement angegeben. Leerzeichen und Zeilenreihenfolge dienen ausschließlich der Übersicht. An nicht explizit aufgeführten Adressen a' sei stets $m(a') = 000\ 000$.

Adresse a	Speicherinhalt $m(a)$
000 000	110 010
001 001	111 111
101 101	010 111
111 111	010 010

Geben Sie in den folgenden Teilaufgaben den durch Ausführung der jeweiligen Operationen resultierenden Speicher m_i jeweils als Tabelle mit Werten im Zweierkomplement an. Adressen a' mit $m_i(a') = 000\ 000$ müssen Sie **nicht** angeben. Heben Sie Zeilen mit $m_i(a) \neq m(a)$ (z.b. farblich) hervor.
Anmerkung: Jede Teilaufgabe beginnt ausgehend von m.

a) $m_1 = memwrite(m,\ 111\ 111,\ memread(m,\ 001\ 001))$
b) $m_2 = memwrite(memwrite(m,\ 000\ 001,\ 010\ 011),\ 000\ 000,\ 000\ 000)$
c) $m_3 = memwrite(m,\ m',\ v)$,mit:
$m' = memread(m,\ 000\ 000) + memread(m,\ 101\ 101)$
$v = memread(m,\ 110\ 110) - 000\ 001$
Geben Sie auch m' und v explizit an.
Hinweis: $a - b = a + (-b)$ *// Anmerkung: siehe Lineare Algebra!*

Lösung 7.1

a)

a	$m_1(a)$
000 000	110 010
001 001	111 111
101 101	010 111
111 111	**111 111**

b)

a	$m_2(a)$
000 001	**010 011**
001 001	111 111
101 101	010 111
111 111	010 010

c)

a	$m_3(a)$
000 000	110 010
001 001	111 111
101 101	010 111
111 111	010 010

In b) ist die Zeile 000 000 nicht mehr aufgeführt, da $m_2(000\ 000) = 000\ 000$.
Für c) gilt:

- $memread(m,\ 000\ 000) + memread(m,\ 101\ 101) = 110\ 010 + 010\ 111 = 001\ 001$
- $v = memread(m,\ 110\ 110) - 000\ 001 = 000\ 000 - 000\ 001$
  $= 000\ 000 + 111\ 111 = 111\ 111$
- keine Zeile ändert sich da, an Adresse 001 001 bereits 111 111 stand.

Aufgabe 7.2 ((0,5 + 0,5 + 1) + (0,5 + 0,5 + 0,5 + 0,5 + 1 + 2 + 2) = 9 Punkte)
Ein *Stapelspeicher* (kurz Stapel) ist eine spezielle Speicherdatenstruktur, die in dieser Aufgabe mit dem in der Vorlesung vorgestellten Speicherkonzept theoretisch modelliert werden soll. Dazu sei $m : \mathbb{N}_0 \to \mathbb{Z} \cup \{\bot\}$ ein Speicher mit unendlich vielen Adressen, in denen jeweils beliebig große ganze Zahlen (als Datenelemente) gespeichert werden können. [*Dieses (sehr unrealistische) Konzept vereinfacht die Modellierung.*] Hierbei repräsentiert $m(a) = \bot$, dass sich an Adresse a im Speicher kein Element befindet.
Der Stapelspeicher sei wie folgt spezifiziert:

- An Adresse 0 soll stets die kleinste Adresse a^*, mit $m(a^*) = \perp$ gespeichert sein. Das bedeutet $m(0) = 1$ gilt genau dann, falls der Speicher „leer“ ist, also keine Elemente enthält. Diese Eigenschaft muss nach jeder *Stapeloperation* erhalten bleiben!

- Datenelemente werden an Adressen ≥ 1 zusammenhängend abgelegt, d.h. das „unterste“ Element befindet sich stets an Adresse 1, das Nächste an Adresse 2, ... bis an Adresse $a^* - 1$ das „oberste“ Element des Stapels liegt.
  Es gilt also stets: $m(a) = \perp$ für genau alle $a \geq a^*$.
  Diese Eigenschaft muss nach jeder *Stapeloperation* erhalten bleiben!

- Die *Stapeloperation* $peek(m)$ soll (falls vorhanden) das oberste Element des Stapels m zurück geben. Falls m leer ist, soll $\perp$ zurück gegeben werden

- Die *Stapeloperation* $push(m, e)$ soll ein neues Datenelement $e \in \mathbb{Z}$ ganz oben in den Stapel m einfügen und den resultierenden Stapel zurück geben.

- Die *Stapeloperation* $pop(m)$ soll (falls vorhanden) genau das oberste Element e eines nicht leeren Stapels m entfernen und den neuen Stapel m' zurück geben.
  pop soll leere Stapel unverändert zurück geben.

Rechts ist ein Stapel tabellarisch dargestellt. Hierbei sei $m_{bsp}(a') = \perp$ für nicht explizit aufgeführte Adressen a'.
Anmerkung: Lassen Sie sich nicht davon irritieren, dass der Stapel „auf dem Kopf“ steht. Wir am KIT sind keine Hochstapler.

Adresse a	Speicherinhalt m(a)
0	3
1	31
2	42
3	$\perp$

Bsp-Stapelspeicher „m_{bsp}“

a) Geben Sie den Stapel $m_1 = push(m_{bsp}, 1337)$ an.
b) Geben Sie den Stapel $m_2 = pop(pop(m_{bsp}))$ an.
c) Geben Sie den Stapel $m_3 = push(pop(m_{bsp}), peek(pop(m_{bsp})))$ an.

Adressen $a > a^*$ brauchen Sie dabei jeweils nicht anzugeben.

Im Folgenden sollen Sie die oben genannten Operationen, sowie einige Hilfsfunktionen, mit Hilfe der in der Vorlesung vorgestellten Speicherfunktionen *memread* und *memwrite* definieren. Beachten Sie dabei die obige Spezifikation des Stapelspeichers. Sie können die Funktionen der anderen Teilaufgaben nutzen, auch wenn Sie diese nicht selbst definiert haben. *Anmerkung: Dies gilt übrigens für alle Aufgaben aller Übungsblätter!*

d) Die Hilfsfunktion $ptr(m)$ soll den Wert an Adresse 0 zurück geben.
e) Die Hilfsfunktion $top(m)$ soll die Adresse des obersten Elements des Stapels m zurückgeben. Gehen Sie davon aus, dass m nicht leer ist.
   Anmerkung: Beachten Sie dies bei der Verwendung von $top(m)$!
f) $inc(m)$ soll einen neuen Speicher m' zurück geben, der sich von m nur an Adresse 0 unterschiedet, wobei $m'(0) = m(0) + 1$ gelten soll. $m(0)$ sei dabei stets definiert.
g) $dec(m)$ soll einen neuen Speicher m' zurück geben, der sich von m nur an Adresse 0 unterschiedet, wobei $m'(0) = m(0) - 1$ gelten soll. $m(0)$ sei dabei stets definiert.
h) Die *Stapeloperation* $peek(m)$ wie oben spezifiziert.
i) Die *Stapeloperation* $push(m, e)$ wie oben spezifiziert.
j) Die *Stapeloperation* $pop(m)$ wie oben spezifiziert.

Lösung 7.2

a)

a	$m_1(a)$
0	4
1	31
2	42
3	1337
4	$\perp$

b)

a	$m_2(a)$
0	1
1	$\perp$

c)

a	$m_1(a)$
0	3
1	31
2	31
3	$\perp$

d) $ptr(m) = memread(m,\ 0)$

e) $top(m) = ptr(m) - 1$

f) $inc(m) = memwrite(m,\ 0,\ ptr(m) + 1)$

g) $dec(m) = memwrite(m,\ 0,\ ptr(m) - 1)$

h) $peek(m) = \begin{cases} memread(m,\ top(m)) & \text{, falls } ptr(m) > 1 \\ \perp & \text{, sonst} \end{cases}$

i) $push(m, e) = inc(memwrite(m,\ ptr(m),\ e))$

j) $pop(m) = \begin{cases} dec(memwrite(m,\ top(m),\ \perp)) & \text{, falls } ptr(m) > 1 \\ m & \text{, sonst} \end{cases}$

Aufgabe 7.3 ((0,5 + 0,5 + 1,5 + 1,5) + (1 + 2 [+ 3]) = 7 [+ 3] Punkte)
Zu einem beliebigen Stapelspeicher m, sei die „effektive Menge“ von m definiert als:

$$M_m = \{(a, m(a)) \mid m(a) \neq \perp\}$$

a) Geben Sie M_{bsp} für den Beispiel-Stapelspeicher m_{bsp} aus dem Aufgabentext der vorigen Aufgabe an.
b) Geben Sie M_{leer} für einen leeren Stapelspeicher m_{leer} an.
c) Geben Sie eine formale Mengendefinition in Abhängigkeit von m für $M_{m'}$, mit $m' = push(m, e)$, mit beliebigem $e \in \mathbb{Z}$ an. *Übersichtshinweise beachten!*
d) Geben sie eine formale Mengendefinition in Abhängigkeit von m für $M_{m'}$ mit $m' = pop(m)$ an. *Übersichtshinweise beachten!*

Verwenden Sie in dieser gesamten Aufgabe zur besseren Übersicht:
- $m(a)$ statt $memread(m, a)$
- $last(m) := m(m(0) - 1)$

Für genau welche Stapel m, bzw. $e \in \mathbb{Z}$ gelten die folgenden Aussagen?

e) $pop(push(m, e)) = m$
Begründen Sie.
f) $push(pop(m), e) = m$
Beweisen Sie, dass Ihre Anforderungen an m notwendig ist sind.
g) Beweisen Sie, dass Ihre Anforderungen an m bzw. e aus Teilaufgabe f) hinreichend sind. *Anmerkung: Diese Teilaufgabe gibt 3 Bonuspunkte.*

Hinweise auf der nächsten Seite beachten!

Sie dürfen hierzu ohne Beweis verwenden, dass gilt:

- $m = m'$ gilt genau für alle Stapel m, m', mit $M_m = M_{m'}$
- $(A \cup B) \setminus B = A$ für disjunkte Mengen A, B
- $(A \setminus B) \cup B = (A \cup B)$ für beliebige Mengen A, B
- $m'(0) = m(0) - 1$ für nichtleere Stapel m und $m' = pop(m)$

Lösung 7.3

a) $M_{Bsp} = \{(0,3),(1,31),(2,42)\}$

b) $M_{leer} = \{(0,1)\}$

c) $M_{push(m,e)} = (M_m \setminus \{(0,\ m(0))\}) \cup \{(0,\ m(0)+1),\ (m(0),\ e)\}$

d) $M_{pop(m)} = \begin{cases} (M_m \setminus \{(0,\ m(0)),\ (m(0)-1,\ last(m))\}) \cup \{(0, m(0)-1)\} & \text{, falls } m(0) > 1 \\ M_m & \text{, sonst} \end{cases}$

e) Dies gilt für jeden beliebigen Stapel m und jedes beliebige $e \in \mathbb{Z}$.
Begründung:
$push(m)$ legt lediglich ein Element e oben auf den Stapel und erhöht den Wert an Adresse 0 genau um 1. $pop(m)$ löscht das oberste Element des Stapels (e) und verringert den Wert an Adresse 0 wieder um 1. Somit ist der Stapel wieder exakt in seinem Ursprungszustand. Da dieses Verhalten nicht von m oder e abhängt, gilt es für beliebige m, e.

f) Für nichtleere m, also formal: m mit $m(0) \geq 2$ und hier genau für $e = last(m)$
Beweis:
Genau für leere Stapel m gilt $m(0) < 2$ und ebenso: $pop(m) = m$ und folglich: $push(pop(m), e) = push(m, e)$, sowie: $M_{push(m,e)} = \{(0,2),\ (1,e)\} \neq \{(0,1)\} = M_m$.
Folglich müssen Stapel nichtleer sein.

g) Für nichtleere Stapel m mit $push(m) = m'$ gilt:

$$M_{m'} = M_{pop(m)} = (M_m \setminus \{(0,\ m(0)),\ (m(0)-1,\ last(m))\}) \cup \{(0, m(0)-1)\}$$

$$M_{push(pop(m),e)} = M_{push(m',e)} = (M_{m'} \setminus \{(0,\ m'(0))\}) \cup \{(0,\ m'(0)+1),\ (m'(0),\ e)\}$$

$$= ([(M_m \setminus \{(0,\ m(0)),\ (m(0)-1,\ last(m))\}) \cup \{(0, m(0)-1)\}] \setminus \{(0,\ m'(0))\}) \cup \{(0,\ m'(0)+1),\ (m'(0),\ e)\}$$

Mit $m'(0) = m(0) - 1$ für $m' = pop(m)$:
$m'(0) = m(0) - 1$
In die obige Formel eingesetzt ergibt das:

$$([(M_m \setminus \{(0,\ m(0)),\ (m(0)-1,\ last(m))\}) \cup \{(0, m(0)-1)\}] \setminus \{(0,\ m'(0))\}) \cup \{(0,\ m'(0)+1),\ (m'(0),\ e)\}$$

$$= ([(M_m \setminus \{(0,\ m(0)),\ (m(0)-1,\ last(m))\}) \cup \{(0, m(0)-1)\}] \setminus \{(0,\ m(0)-1)\}) \cup \{(0,\ m(0)),\ (m(0)-1,\ e)\}$$

Da M_m nur ein Element $(0, m(0))$ enthält, weil m als Funktion rechtseindeutig ist, gilt Disjunktheit zwischen $(M_m \setminus \{(0,\ m(0)),\ (m(0)-1,\ last(m))\})$ und $\{(0,\ m(0)-1)\}$ und somit:

$$([(M_m \setminus \{(0,\ m(0)),\ (m(0)-1,\ last(m))\}) \cup \{(0, m(0)-1)\}] \setminus \{(0,\ m(0)-1)\}) \cup \{(0,\ m(0)),\ (m(0)-1,\ e)\}$$

$$= (M_m \setminus \{(0,\ m(0)),\ (m(0)-1,\ last(m))\}) \cup \{(0,\ m(0)),\ (m(0)-1,\ e)\}$$

$$= M_m, \qquad \text{genau dann, wenn:} \quad last(m) = e$$

Aufgabe 7.4 (0 Punkte)

Bei Dr. Meta stapeln sich die Daten von seinen B.I.R.D-Drohnen. Um sie 0effizient abzuarbeiten hat er beschlossen in einen mega intelligenten maschinellen Arbeitsdruiden zu investieren. Leider sind diese Hochleistungs-Quanten-KIs (sehr)^{31} teuer. Daher hat er sich diesen Kaufwunsch extra aufgehoben, in der Hoffnung am schwarzen Freitag ein super Schnäppchen zu erhaschen. Da dieser zufällig auf ihren freien Tag fiel, hat er seine Ersatzassistentin Regina Einfall beauftragt, nach den besten Angeboten Ausschau zu halten. Angeblich hat Regina einen absoluten Hammer-Deal auf Wunsch.de ergattert und gleich 1336,99 dieser Modelle für einen geradezu surreal geringen Preis erstanden. Allerdings zieht sich die gratis „yesterday delivery“^{TM} noch etwas hin. Bei der Supporthotline des Verkäufers ist seit Tagen alles belegt, aber der nächste freie Mitarbeiter, so versichert die Stimme in der Warteschleife unermüdlich, ist ganz bestimmt für Sie reserviert!

b) Warten Sie gespannt, wie sich die weitere Lage entwickelt.