Prof. Dr. J. Giesl
S. Dollase, M. Hark

Aufgabe 2 (Programmierung mit Arrays — VPL): (7 Punkte)

Selectionsort ist ein Algorithmus zum Sortieren von Arrays, der wie folgt vorgeht, um ein Array `a` zu sortieren: Das Array wird wiederholt von links nach rechts durchlaufen. Am Ende des $(n-1)$-ten Durchlaufs gilt, dass die ersten $(n-1)$ Array-Elemente an ihrer endgültigen Position stehen, d.h. bis zur Stelle $n-2$ ist das Array bereits korrekt sortiert. Folglich müssen im n-ten Durchlauf nur noch die letzten `a.length`$-n+1$ Elemente betrachtet werden. Im n-ten Durchlauf wird die Position i_{min} des kleinsten Elementes der letzten `a.length`$-n+1$ Elemente bestimmt. Danach wird das Element an Position $n-1$ mit dem Minimum an Position i_{min} vertauscht.
Als Beispiel betrachten wir das Array `{3,7,1,5,8}`.
Im ersten Durchlauf $(n=1)$ wird $i_{min}=2$ bestimmt, da die kleinste Zahl (1) des Arrays an Position 2 steht. Sie wird daher mit dem Element 3 an Position $n-1=0$ vertauscht und es entsteht das Array `{1,7,3,5,8}`.
Im zweiten Durchlauf $(n=2)$ bestimmt man $i_{min}=2$ und erhält `{1,3,7,5,8}`.
Im dritten Durchlauf ergibt sich $i_{min}=3$, dies führt zu `{1,3,5,7,8}`.
Im vierten und letzten Durchlauf $(n=4)$ wird wieder $i_{min}=3$ bestimmt. Die Vertauschung des Elements an der Position $n-1=3$ mit dem Element an der Position $i_{min}=3$ ändert daher nichts an dem Array. Das sortierte Array ist also `{1,3,5,7,8}`.
Implementieren Sie eine Klasse `Selection` mit einer Methode `public static void selection(int[] a)`, die das Array `a` mithilfe des Algorithmus *Selectionsort* aufsteigend sortiert.

Hinweise:

- Im `RWTHmoodle`-Lernraum "Programmierung (Übung - Tutorium)" stehen drei `Java`-Dateien `Sort.java`, `Bubble.java` und `Selection.java` zum Download zur Verfügung. Speichern Sie alle drei Dateien in einem neuen Ordner. Die Klasse `Selection` enthält eine Methode `selection` mit leerem Rumpf. Wenn Sie die Implementierung vervollständigen und anschließend mit `javac Sort.java` kompilieren, dann können Sie Ihre Implementierung mit `java Sort selection` testen.

  Zusätzlich dazu können Sie Ihre Implementierung direkt in `VPL` testen. Klicken Sie dazu einfach auf das Raketensymbol. Anschließend wird Ihr Code in `VPL` kompiliert und mit der in `Sort.java` beschriebenen Testmethode getestet.

Aufgabe 4 (Verifikation mit Arrays): (8 + 2 = 10 Punkte)

Gegeben sei folgendes Java-Programm P:

$\langle$ `true` $\rangle$ (Vorbedingung)

```
i = 0;
res = false;
while(i < a.length) {
  if(x == a[i]) {
    res = true;
  }
  i = i + 1;
}
```

$\langle$ `res` $=$ `x` $\in \{$`a`$[j] \mid 0 \leq j \leq$ `a.length`$-1\}$ $\rangle$ (Nachbedingung)

a) Vervollständigen Sie die folgende Verifikation des Algorithmus im Hoare-Kalkül, indem Sie die unterstrichenen Teile ergänzen. Hierbei dürfen zwei Zusicherungen nur dann direkt untereinander stehen, wenn die untere aus der oberen folgt. Hinter einer Programmanweisung darf nur eine Zusicherung stehen, wenn dies aus einer Regel des Hoare-Kalküls folgt.

Beachten Sie bei der Anwendung der "Bedingungsregel 1" mit Vorbedingung φ und Nachbedingung ψ, dass $\varphi \wedge \neg B \implies \psi$ gelten muss. D. h. die Nachbedingung ψ der `if`-Anweisung muss aus der Vorbedingung φ der `if`-Anweisung und der negierten Bedingung $\neg B$ folgen. Geben Sie beim Verwenden der Regel einen entsprechenden Beweis an.

Hinweise:

- Sie dürfen beliebig viele Zusicherungs-Zeilen ergänzen oder streichen. In der Musterlösung werden allerdings genau die angegebenen Zusicherungen benutzt.
- Bedenken Sie, dass die Regeln des Kalküls syntaktisch sind, weshalb Sie semantische Änderungen (beispielsweise von $x+1 = y+1$ zu $x = y$) nur unter Zuhilfenahme der Konsequenzregeln vornehmen dürfen.
- Der Ausdruck `x` $\in$ `M` hat den Wert `true`, wenn `x` in der Menge `M` enthalten ist, sonst hat der Ausdruck den Wert `false`.

$\langle$`true`$\rangle$

$\langle$_____$\rangle$

```
i = 0;
```

$\langle$_____$\rangle$

```
res = false;
```

$\langle$_____$\rangle$

$\langle$_____$\rangle$

```
while (i < a.length) {
```

$\langle$_____$\rangle$

```
    if (x == a[i]) {
```

$\langle$_____$\rangle$

$\langle$_____$\rangle$

```
        res = true;
```

$\langle$_____$\rangle$

```
    }
```

$\langle$_____$\rangle$

```
    i = i + 1;
```

$\langle$_____$\rangle$

```
}
```

$\langle$_____$\rangle$

$\langle$`res` $=$ `x` $\in \{$`a`$[j] \mid 0 \leq j \leq$ `a.length` $-1\}\rangle$

b) Untersuchen Sie den Algorithmus P auf seine Terminierung. Für einen Beweis der Terminierung muss eine Variante angegeben werden und unter Verwendung des Hoare-Kalküls die Terminierung bewiesen werden.

Geben Sie auch bei dieser Teilaufgabe einen Beweis für die Aussage $\varphi \wedge \neg B \implies \psi$ bei der Anwendung der "Bedingungsregel 1" an.

Aufgabe 5 (Deck 2): (Codescape)

Lösen Sie die Räume von Deck 2 des Spiels Codescape.
Ihre Lösung für Räume dieses Codescape Decks wird nur dann für die Zulassung gezählt, wenn Sie die Lösung bis Montag, den 11.11.2019, um 12:00 Uhr abschicken.