Test 1 in Programmkonstruktion

37 / 40 Punkte

Alle Aufgaben beziehen sich auf Java.

1. Single-Choice Fragen zu Datenstrukturen

15 / 15 Punkte

```
Deque<String> deque = new LinkedList<String>();
Map<Integer,String> map = new HashMap<Integer,String>();
deque.offerFirst("a");
deque.offer("b");
deque.offerFirst("c");
map.put(1, deque.poll());
map.put(2, deque.peek());
String x = map.get(3);
deque.offer(x);
String y = deque.poll();
```

Welche Aussagen sind nach Ausführung der obigen Anweisungen korrekt?

Aufgabe 1.1.

3 / 3 Punkte

`map.get(1)` liefert

- ○ null
- ○ a
- ○ b
- ● c
- ○ einen anderen Wert
- ○ einen Laufzeitfehler

Aufgabe 1.2.

3 / 3 Punkte

`deque.poll()` liefert

- ○ null
- ○ a
- ● b
- ○ c
- ○ einen anderen Wert
- ○ einen Laufzeitfehler

Aufgabe 1.3.

3 / 3 Punkte

`y` liefert

- ○ null
- ● a
- ○ b
- ○ c
- ○ einen anderen Wert
- ○ einen Laufzeitfehler

Aufgabe 1.4.

3 / 3 Punkte

`map.get(3)` liefert

- [x] null
- [ ] a
- [ ] b
- [ ] c
- [ ] einen anderen Wert
- [ ] einen Laufzeitfehler

Aufgabe 1.5.

3 / 3 Punkte

`map.get(2)` liefert

- [ ] null
- [x] a
- [ ] b
- [ ] c
- [ ] einen anderen Wert
- [ ] einen Laufzeitfehler

2. Single-Choice Fragen zu Arrays und Rekursion

12 / 15 Punkte

Folgende Implementierungen der Methode `min` sind syntaktisch korrekt. Die Methode soll den kleinsten Eintrag im übergebenen Array ab (inklusive) Index `i` aufwärts zurückgeben. Vorbedingungen: `i >= 0` und `i < xs.length`. Geben Sie an, welche Aussage auf die jeweilige Implementierung zutrifft.

Aufgabe 2.1.

0 / 3 Punkte

```
public static int min(int[] xs, int i) {
    int m = xs[i];
    int j = xs.length;
    while (--j > i) {
        int x = xs[j];
        m = x < m ? x : m;
    }
    return m;
}
```

Welche Aussage trifft hier zu?

- [ ] Laufzeitfehler (erzeugt einen Laufzeitfehler bei bestimmten gültigen Eingaben)
- [x] falscher Wert (bei allen gültigen Eingaben kein Laufzeitfehler aber liefert einen falschen Wert bei bestimmten gültigen Eingaben)
- [ ] korrekt (liefert für alle gültigen Argumente das korrekte Ergebnis)

Aufgabe 2.2.

3 / 3 Punkte

```
public static int min(int[] xs, int i) {
    if (i == xs.length - 1) {
        return xs[i];
    } else if (xs[i] < min(xs, i+1)) {
        return xs[i];
    } else {
        return min(xs, i+1);
    }
}
```

Welche Aussage trifft hier zu?

- ○ Laufzeitfehler
- ○ falscher Wert
- ● korrekt

Aufgabe 2.3.

3 / 3 Punkte

```
public static int min(int[] xs, int i) {
    if (i == xs.length - 1) {
        return xs[i];
    }
    int x = min(xs, i+1);
    int y = xs[i];
    return x > y ? x : y;
}
```

Welche Aussage trifft hier zu?

- ○ Laufzeitfehler
- ● falscher Wert
- ○ korrekt

Aufgabe 2.4.

3 / 3 Punkte

```
public static int min(int[] xs, int i) {
    int x = min(xs, i+1);
    if (i < xs.length - 1) {
        return x < xs[i] ? x : xs[i];
    } else {
        return xs[i];
    }
}
```

Welche Aussage trifft hier zu?

- [x] Laufzeitfehler
- [ ] falscher Wert
- [ ] korrekt

Aufgabe 2.5.

3 / 3 Punkte

```
public static int min(int[] xs, int i) {
    if (i == xs.length) {
        return xs[i];
    } else {
        return xs[i] < min(xs, i+1) ? xs[i] : min(xs, i+1);
    }
}
```

Welche Aussage trifft hier zu?

- [x] Laufzeitfehler
- [ ] falscher Wert
- [ ] korrekt

3. Auswahlaufgaben

10 / 10 Punkte

In den Methoden sind die Buchstaben A, B, C und D jeweils durch einen der vorgeschlagenen Programmteile zu ersetzen. Bitte wählen Sie für jeden dieser Buchstaben genau eine zutreffende Antwortmöglichkeit. Die Methoden müssen sich so verhalten, wie in den Kommentaren angegeben. Punkte gibt es nur, wenn die gewählten Antwortmöglichkeiten zusammenpassen.

Aufgabe 3.1.

5 / 5 Punkte

```
// reverse dreht die Reihenfolge der Elemente der übergebenen Queue um.
// Vorbedingung: xs ist nicht null.
//
// Beispiel:
//
//     Queue<Integer> xs = new LinkedList<Integer>();
//     xs.offer(1); xs.offer(2); xs.offer(3); xs.offer(4);
//     for (int x : xs) System.out.print(x);
//     System.out.println();
//     reverse(xs);
//     for (int x : xs) System.out.print(x);
//
// gibt folgendes aus:
//
//     1234
//     4321
//
public static void reverse(Queue<Integer> xs) {
    Integer x = A;
    if (x != B) {
        xs.poll();
        C;
        D;
    }
}
```

A:

- ○ `reverse(xs)`
- ○ `null`
- ● `xs.peek()`
- ○ `xs.poll()`

B:

- ○ `xs.poll()`
- ○ `reverse(xs)`
- ● `null`
- ○ `xs.offer(x)`

C:

- ○ `xs.offer(x)`
- ● `reverse(xs)`
- ○ `xs.peek()`
- ○ `xs.poll()`

D:

- [x] `xs.offer(x)`
- [ ] `reverse(xs)`
- [ ] `xs.peek()`
- [ ] `xs.poll()`

Aufgabe 3.2.

5 / 5 Punkte

```
// product berechnet das Produkt aller double-Werte im übergebenen Array.
// Vorbedingung: m != null und kein Eintrag in m ist null.
public static double product(double[][] m) {
    return go(m,0,0);
}

public static double go(double[][] m, int i, int j) {
    if (i == m.length) {
        return A;
    } else if (j == A) {
        return m[i][j] * C;
    } else {
        return m[i][j] * D;
    }
}
```

A:

- [ ] `0`
- [ ] `go(m,i+1,0)`
- [ ] `m[0][0]`
- [ ] `m[i][j]`
- [x] `1`

B:

- [ ] `0`
- [ ] `m[i].length`
- [ ] `i`
- [x] `m[i].length - 1`
- [ ] `m.length`
- [ ] `m.length - 1`

C:

- [ ] `go(m,0,j+1)`
- [x] `go(m,i+1,0)`
- [ ] `go(m,i,j+1)`
- [ ] `go(m,i+1,j)`

D:

- `go(m,0,j+1)`
- `go(m,i+1,0)`
- **(selected)** `go(m,i,j+1)`
- `go(m,i+1,j)`