252-0027-00: Einführung in die Programmierung

Lösungen zur Theorie-Probeprüfung (Teil I)

Aufgabe 1 (3) -2 pro Fehler

1) `System.out.println(7 + 5 / 4 + (2 + "==") + 5 * 2);`

   82 == 10

2) `System.out.println(6 / 3 % 6 + (double) 5 / 4 + 3 / 4);`

   3.25

3) `System.out.println(false != !false && ( 12/4 ) > 2 && (12 / (2 / 4)) == -1);`

   java.lang.ArithmeticException: / by zero

Aufgabe 2 (5) -3 pro Fehler

1) `System.out.println(24 / (2 * 12 / 4));`

   // Output: 4

2) `System.out.println(72 % (5 * 9 +| * |/ 1) / 4);`

   // Output: 6

3) `System.out.println(36 / (4 − 10));`

   // Output: −6

Aufgabe 3 (8) -2 pro Fehler

Gegeben sei in Abbildung 1 die EBNF-Beschreibung von expr.

digit ⇐ `0` | `1` | ... | `9`
char ⇐ `A` | `B` | ... | `Z` | `a` | `b` | ... | `z`
num ⇐ *digit* { *digit* } [`.` *digit* { *digit* }]
var ⇐ *char* { *char* }
func ⇐ *char* { *char* } `(`
op ⇐ `+` | `-` | `*` | `/` | `^`
open ⇐ `(`
close ⇐ `)`

atom ⇐ *num* | *var*
term ⇐ *open expr close* | *func expr close* | *atom*
expr ⇐ *term* [*op term*]

Abbildung 1: **EBNF-Beschreibung** von *expr*

Geben Sie für jeden folgenden Ausdruck an, ob er nach der EBNF-Beschreibung von expr in Abbildung 1 gültig ist.

Ausdruck	**Gültig**	**Ungültig**	**Ausdruck**	**Gültig**	**Ungültig**
1337	X		(o)^(o)	X	
x = 5		X	10.5 / 0	X	
y + z - 7		X	x + x(1 - x)	X	
sin(cos(sin(x)))	X		(x0 + x1)		X
sqrt(-4 + 20)		X	fun1(3.0 * q)		X

Aufgabe 4 (4) -2 pro Fehler

Gegeben sind die Precondition und Postcondition für das folgende Programm.

```
public int compute(int n, int m) {
     // Precondition: m >= n && n >= 0
     int x = n;
     int y = m;

     while (x<y){
       x = x + 1;
       y = y - 1;
     }

     // Postcondition: y == (n+m)/2
     return y;
}
```

Bitte geben Sie die Loop Invariante als Java Expression an.

Loop Invariante: $x + y == n + m \quad \&\& \quad n + m >= 0 \quad \&\& \quad x - y \leq 1$
oder: $x + y == n + m \quad \&\& \quad x + y >= 0 \quad \&\& \quad x \leq y + 1$
oder: $x + y == n + m \quad \&\& \quad n + m >= 0 \quad \&\& \quad x - 1 \leq y$

Aufgabe 5 (8) -2 pro Fehler

Gegeben sei die Methode maxEven(int[]) einer Klasse. In dieser Aufgabe stehen { und } – die Klammern für einen Block – auf einer separaten Zeile. Welche dieser Statements können entfernt werden ohne dass das Ergebnis der Methode verändert wird – egal, was für Parameter der Methodeübergeben werden?

Zum Beispiel sind im Codesegment

```
if (true) {     // Redundant
     x++;
} else {        // Redundant
     x--;        // Redundant
}
```

```
   public static int maxEven(int[] arr){      // Platz fuer Antwort

     int max = -1;                          //

     for (int i=0; i<arr.length; i++) {     //

          if(arr[i] % 2 == 0) {             //
               if ( max == -1 ) {           //
                    max = arr[i];           //
               }
               if ( arr[i] > max ) {        //
                    max = arr[i];           //
               }
               if (arr[i] == max) {         //R
                    max = arr[i];           //R
               }
          } else {                          //R
               if ( max % 2 != 0) {         //R
                    max = -1;               //R
               }
          }
     }

     if (max >= 0) {                        //R
          return max;                       //(R)

     } else {                               //R
          return max++;                     //(R)
     }
}
```

Alle blauen R müssen vorhanden sein. Genau eines der beiden orangen (R) muss vorhanden sein.

Aufgabe 6 (4) -2 pro Fehler

Bitte geben Sie für die folgenden Programmsegmente die schwächste Vorbedingung (weakest pre-condition) an. Bitte verwenden Sie Java Syntax (die Klammern { und } können Sie weglassen).

1) P: $\{2 * x * z + 1 > 0 \ \&\& \ 2 * x < 5\} = \{x * z >= 0 \ \&\& \ x <= 2\}$
```
y = 2 * x;
z = z * y + 1;
Q: { z > 0 && y < 5 }
```

2) P: `{ a > b && (a-b)*(b-a)>0 || a <= b && b*b > 0} = { a <= b && b*b > 0}`
```
if (a > b) {
  c = (a-b)*(b-a);
} else {
  c = b*b;
}
Q: { c > 0 }
```

Aufgabe 7 (8) -2 pro Fehler

1) Welche dieser Hoare-Tripel sind (un)gültig? Bitte geben Sie für ungültige Tripel ein Gegenbeispiel an. Das Gegenbeispiel muss Werte für Variable angeben, so dass nach Ausführung der Anweisung(en) die Postcondition nicht gilt. Alle Anweisungen sind Teil einer Java Methode. Alle Variablen sind vom Typ int und es gibt keinen Overflow.

   a) `{ x > 0 } y = x / 2; { y > 0 }`
   ungültig: x = 1 ⇒ y = 0

   b) `{ true } a = b / 2; a--; { a < b }`
   ungültig: b = -2 ⇒ a = -2

   c)
```
{ x > 0 || y > 0 }
if (x > y) {
  z = x-y;
} else {
  z = y-x;
}
{ z > 0 }
```
   ungültig: x = 1 y = 1 ⇒ z = 0

2) Gegeben sind jeweils ein Programm zusammen mit einer Precondition und einer Postcondition. Auf der rechten Seite finden Sie jeweils drei potentielle Invarianten für die Schleife im Programm. Höchstens eine Option davon ist eine gültige Invariante. Sofern eine davon gültig ist, kreuzen Sie das dazugehörige Feld an. Sollte keine davon gültig sein, kreuzen Sie die vierte Option an.

a)
```
P: { x >= 0 }
int i = 0;
int s = 0;
while(i <= x) {
  s = s + i;
  i = i + 1;
}
Q: { s == x*(x+1)/2 }
```

Wählen Sie die passende Option:

☐ `{i <= x && s == (i-1)*i/2}`
☐ `{i <= x && s == i*(i+1)/2}`
☒ `{i <= x + 1 && s == (i-1)*i/2}`
☐ Keine der drei Optionen ist eine gültige Invariante

b)
```
P: { n >= 0 }
int y = 0;
int z = 0;
while(y*y < n) {
  y = y + 1;
  if (y*y <= n) {
    z = z + 1;
  }
}
Q: {z == Math.floor(Math.sqrt(n))}
```

Wählen Sie die passende Option:

☐ `{y*y <= n && z*z <= n}`
☐ `{y*y <= n && (y == z || y == z+1)}`
☐ `{z*z <= n && (y == z || y == z+1)}`
☒ Keine der drei Optionen ist eine gültige Invariante

`{z*z <= n && (y*y <= n && y == z || y*y > n && y == z+1)}`