SoSe 2020

Prof. Dr. Margarita Esponda

Objektorientierte Programmierung

5. Übungsblatt

Lernziel: Zusicherungen und Programmverifikation.

1. Aufgabe (6 Punkte)

Beweisen Sie die Gültigkeit der folgenden Programmformeln.

$$\{P\} \equiv \{a > 0 \ \wedge \ b > 0 \ \wedge \ c < 0\}$$

```
a = a + b - c
d = b
b = a - b - c
c = - c
```

$$\{Q\} \equiv \{a > 0 \ \wedge \ b > 0 \ \wedge \ c > 0 \ \wedge \ b == a - d + c\}$$

2. Aufgabe (8 Punkte)

Seien **x, y, z** ganzzahlige Variablen (**int**) und **c** ein konstanter ganzer Wert größer 0.

Beweisen Sie die Gültigkeit der folgenden Programmformel.

$$\{P\} \equiv \{x \geq 0 \wedge (x * y + z) == c\}$$

```
if x%2==0:
    y = y + y
    x = x//2
else:
    z = z + y
    x = x - 1
```

$$\{Q\} \equiv \{x \geq 0 \wedge (x * y + z) == c\}$$

3. Aufgabe (8 Punkte)

Beweisen Sie die Gültigkeit der Programmformel (1).

$$\{P \equiv b > 0 \wedge a > 0\}$$ .....

```
counter = 1                                                    .
power = b                                                      .
```

$$\{INV \equiv (b > 0) \wedge (power == b**counter) \wedge (a \geq counter \geq 0)\}$$.

```
while counter<a:                                               .
    power = power * b                                          .
    counter = counter + 1                                      .
```

$$\{Q \equiv power == b**a\}$$ .....(1)

4. Aufgabe (6 Punkte)

Gegeben sei folgendes Python-Programm, das die Anzahl der Bits ohne führende Nullen von einer positiven Zahl berechnet und in der Variablen **z** ablegt.

$$\{P\} \equiv \{Zahl \geq 0\}$$

```
hilf = Zahl
bits = 1
while hilf>1:
    hilf = hilf//2
    bits = bits+1
```

$$\{Q\} \equiv \{bits == BinaryDigits(Zahl)\}$$

Dabei sei $BinaryDigits(z)$ die Anzahl der Bits der Binärdarstellung der Zahl **z** ohne führende Nullen.

Beispiel: $BinaryDigits(6)$ ergibt 3

a) Welche der folgenden Prädikate stellen eine geeignete Invariante der **while**-Schleife dar?

- ☐ $hilf \geq 0 \wedge (BinaryDigits(hilf) + bits == BinaryDigits(Zahl) + 1)$
- ☐ $hilf \geq 0 \wedge (BinaryDigits(hilf) + 1 == BinaryDigits(Zahl) + bits)$
- ☐ $hilf \geq 0 \wedge (BinaryDigits(hilf) + bits == BinaryDigits(Zahl) + bits)$

b) Beweisen Sie die **Invarianten**-Eigenschaft für das von Ihnen unter a) genannte Prädikat. Sie können benutzen, dass für jede positive ganze Zahl ***z*** folgendes gilt:

$$BinaryDigits(z) = \begin{cases} 1, & falls \quad z = 0 \quad oder \quad z = 1 \\ BinaryDigits(z\,//\,2) + 1, & falls \quad z > 1 \end{cases}$$

5. Aufgabe (6 Punkte)

a) Ersetzen Sie die **for**-Schleife des **Insertsort**-Algorithmus der Vorlesung mit einer **while**-Schleife. Postulieren Sie aussagekräftige Invarianten für die äußeren und inneren Schleifen.

b) Testen Sie Ihre Invarianten mit Hilfe von **assert**-Anweisungen. Um Ihre Invarianten zu testen, dürfen Sie Hilfsfunktionen verwenden.

6. Aufgabe (**6 Bonuspunkte**)

a) Was ist die wichtigste Invariante der Schleife innerhalb folgender **partition**-Funktion?

```
def partition( A, low, high ):
    pivot = A[low]
    i = low
    for j in range(low+1,high+1):
        if ( A[j] < pivot ):
            i=i+1
            A[i], A[j] = A[j], A[i]
    A[i], A[low] = A[low], A[i]
    return i
```

b) Begründen Sie Ihre Antwort.

Wichtige Hinweise:

1) In allen Beweisen müssen Sie die jeweils verwendeten Hoare-Regeln angeben.
2) Jeder Beweisschritt muss begründet werden.
3) Schreiben Sie bei Bedarf Kommentare, die den Tutoren die Korrektur der Lösungen erleichtern können.