Suggested Solutions for Homework Assignment #4

We assume the binding powers of the logical connectives and the entailment symbol decrease in this order: $\neg$, $\{\forall, \exists\}$, $\{\wedge, \vee\}$, $\rightarrow$, $\leftrightarrow$, $\vdash$.

1. Prove that the following annotated program segments are correct:

   (a) (10 points)

   $\{true\}$
   if $x < y$ **then** $x, y := y, x$ **fi**
   $\{x \geq y\}$

   Solution.

$$\dfrac{\dfrac{\dfrac{\text{pred. calculus + algebra}}{true \wedge x < y \rightarrow y \geq x} \qquad \dfrac{}{\{\ y \geq x\ \}\ x, y := y, x\ \{\ x \geq y\ \}}\ (\text{Assign})}{\{\ true \wedge x < y\ \}\ x, y := y, x\ \{\ x \geq y\ \}}\ (\text{SP}) \qquad \dfrac{\text{pred. calculus + algebra}}{true \wedge \neg(x < y) \rightarrow x \geq y}}{\{\ true\ \}\ \textbf{if}\ x < y\ \textbf{then}\ x, y := y, x\ \textbf{fi}\ \{\ x \geq y\ \}}\ (\text{If-Then})$$

   □

   (b) (10 points)

   $\{g = 0 \wedge p = n \wedge n \geq 1\}$
   while $p \geq 2$ **do**
   $\quad g, p := g + 1, p - 1$
   od
   $\{g = n - 1\}$

   Solution.

$$\dfrac{\dfrac{\text{pred. calculus + algebra}}{g = 0 \wedge p = n \wedge n = 1 \rightarrow p > 0 \wedge p + g = n} \qquad \alpha \qquad \dfrac{\text{pred. calculus + algebra}}{p > 0 \wedge p + g = n \wedge \neg(p \geq 2) \rightarrow g = n - 1}}{\{\ g = 0 \wedge p = n \wedge n = 1\ \}\ \textbf{while}\ p \geq 2\ \textbf{do}\ g, p := g - 1, p + 1\ \textbf{od}\ \{\ g = n - 1\ \}}\ (\text{Consequence})$$

   α :

$$\dfrac{\dfrac{\beta \qquad \dfrac{}{\{\ p + 1 > 0 \wedge (p + 1) + (g - 1) = n\ \}\ g, p := g - 1, p + 1\ \{\ p > 0 \wedge p + g = n\ \}}\ (\text{Assign})}{\{\ p > 0 \wedge p + g = n \wedge p \geq 2\ \}\ g, p := g - 1, p + 1\ \{\ p > 0 \wedge p + g = n\ \}}\ (\text{SP})}{\{\ p > 0 \wedge p + g = n\ \}\ \textbf{while}\ p \geq 2\ \textbf{do}\ g, p := g - 1, p + 1\ \textbf{od}\ \{\ p > 0 \wedge p + g = n \wedge \neg(p \geq 2)\ \}}\ (\text{while})$$

   β :

$$\dfrac{\text{pred. calculus + algebra}}{p > 0 \wedge p + g = n \wedge p \geq 2 \rightarrow p + 1 > 0 \wedge (p + 1) + (g - 1) = n}$$

   □

   (c) (20 points) For this program, prove its total correctness.

$\{y > 0 \land (x \equiv m \pmod y)\}$
while $x \geq y$ **do**
    $x := x - y$
od
$\{(x \equiv m \pmod y) \land x < y\}$

Solution.

$$\dfrac{\alpha \qquad \dfrac{\text{pred. calculus + algebra}}{y > 0 \land (x \equiv m \pmod y) \land \neg(x \geq y) \to (x \equiv m \pmod y) \land x < y}}{\{\ y > 0 \land (x \equiv m \pmod y)\ \}\ \textbf{while}\ x \geq y\ \textbf{do}\ x := x - y\ \textbf{od}\ \{\ (x \equiv m \pmod y) \land x < y\ \}}\ (\text{WP})$$

α :

$$\dfrac{\beta \qquad \gamma \qquad \dfrac{\text{pred. calculus + algebra}}{y > 0 \land (x \equiv m \pmod y) \land x \geq y \to x \geq 0}}{\begin{array}{c}\{\ y > 0 \land (x \equiv m \pmod y)\ \} \\ \textbf{while}\ x \geq y\ \textbf{do}\ x := x - y\ \textbf{od} \\ \{\ y > 0 \land (x \equiv m \pmod y) \land \neg(x \geq y)\ \}\end{array}}\ (\text{while: simply total})$$

β :

$$\dfrac{\dfrac{\text{pred. calculus + algebra}}{\begin{array}{c} y > 0 \land (x \equiv m \pmod y) \land x \geq y \to \\ y > 0 \land ((x - y) \equiv m \pmod y)\end{array}} \qquad \begin{array}{c}\dfrac{}{\{\ y > 0 \land ((x - y) \equiv m \pmod y)\ \}}\ (\text{Assign}) \\ x := x - y \\ \{\ y > 0 \land (x \equiv m \pmod y)\ \}\end{array}}{\{\ y > 0 \land (x \equiv m \pmod y) \land x \geq y\ \}\ x := x - y\ \{\ y > 0 \land (x \equiv m \pmod y)\ \}}\ (\text{SP})$$

γ :

$$\dfrac{\dfrac{\text{pred. calculus + algebra}}{y > 0 \land (x \equiv m \pmod y) \land x \geq y \land x = Z \to x - y < Z} \qquad \dfrac{}{\{\ x - y < Z\ \}\ x := x - y\ \{\ x < Z\ \}}\ (\text{Assign})}{\{\ y > 0 \land (x \equiv m \pmod y) \land x \geq y \land x = Z\ \}\ x := x - y\ \{\ x < Z\ \}}\ (\text{SP})$$

□

2. (30 %) Below is a program that finds the minimum and the maximum elements of an array of n (assumed to be positive and even) integers. The elements of an array are indexed from 1 through n.

```
if (a[1] < a[2]) then
  min := a[1];
  max := a[2]
else
  min := a[2];
  max := a[1]
fi;

i := 3;
while (i<=n) do
  if (a[i] < a[i+1]) then
    if (a[i] < min) then
      min := a[i]
    fi;
    if (a[i+1] > max) then
```

```
      max := a[i+1];
    fi
  else
    if (a[i+1] < min) then
      min := a[i+1]
    fi;
    if (a[i] > max) then
      max := a[i]
    fi
  fi;
  i := i + 2;
od;
```

Annotate the program into a *standard* proof outline, showing clearly the partial correctness of the program; a standard proof outline is essentially an annotated program where every statement is preceded by a pre-condition and the entire program is followed by a post-condition.

Solution. Let $isMin(m, a, i)$ denote that m is the minimum element in $a[1..i]$ and $isMax(M, a, i)$ denote that M is the maximum element in $a[1..i]$. In particular, $isMin(\_, a, 0)$ and $isMax(\_, a, 0)$ both hold, as $a[1..0]$ denotes the empty array. Let $odd(i)$ denote that i is odd.

```
// assume n is positive and even, which is preserved by the code and
// will be omitted later
if (a[1] < a[2]) then
// isMin(a[1], a, 2) ∧ isMax(a[2], a, 2)
  min := a[1];
// isMin(min, a, 2) ∧ isMax(a[2], a, 2)
  max := a[2]
else
// isMin(a[2], a, 2) ∧ isMax(a[1], a, 2)
  min := a[2];
// isMin(min, a, 2) ∧ isMax(a[1], a, 2)
  max := a[1]
fi;
// isMin(min, a, 2) ∧ isMax(max, a, 2)

i := 3;
// inv: (3 ≤ i ≤ n + 1) ∧ odd(i) ∧ isMin(min, a, i − 1) ∧ isMax(max, a, i − 1)
while (i<=n) do
  // (3 ≤ i ≤ n) ∧ odd(i) ∧ isMin(min, a, i − 1) ∧ isMax(max, a, i − 1)
  if (a[i] < a[i+1]) then
    // (3 ≤ i ≤ n)∧odd(i)∧(a[i] < a[i+1])∧isMin(min, a, i−1)∧isMax(max, a, i−1)
    if (a[i] < min) then
    // (3 ≤ i ≤ n)∧odd(i)∧(a[i] < a[i+1])∧isMin(a[i], a, i+1)∧isMax(max, a, i−1)
      min := a[i]
    fi;
    // (3 ≤ i ≤ n)∧odd(i)∧(a[i] < a[i+1])∧isMin(min, a, i+1)∧isMax(max, a, i−1)
    if (a[i+1] > max) then
```

```
          // (3 ≤ i ≤ n) ∧ odd(i) ∧ (a[i] < a[i + 1]) ∧ isMin(min, a, i + 1)∧
          // isMax(a[i + 1], a, i + 1)
          max := a[i+1];
        fi
    else
        // (3 ≤ i ≤ n)∧odd(i)∧(a[i] ≥ a[i+1])∧isMin(min, a, i−1)∧isMax(max, a, i−1)
        if (a[i+1] < min) then
          // (3 ≤ i ≤ n) ∧ odd(i) ∧ (a[i] ≥ a[i + 1]) ∧ isMin(a[i + 1], a, i + 1)∧
          // isMax(max, a, i − 1)
          min := a[i+1]
        fi;
        // (3 ≤ i ≤ n)∧odd(i)∧(a[i] ≥ a[i+1])∧isMin(min, a, i+1)∧isMax(max, a, i−1)
        if (a[i] > max) then
          // (3 ≤ i ≤ n) ∧ odd(i) ∧ (a[i] ≥ a[i + 1]) ∧ isMin(min, a, i + 1)∧
          // isMax(a[i], a, i + 1)
          max := a[i]
        fi
    fi;
    // (3 ≤ i ≤ n) ∧ odd(i) ∧ isMin(min, a, i + 1) ∧ isMax(max, a, i + 1)
    i := i + 2;
    // (3 ≤ i ≤ n + 1) ∧ odd(i) ∧ isMin(min, a, i − 1) ∧ isMax(max, a, i − 1)
od;
// isMin(min, a, n) ∧ isMax(max, a, n)
```

□

3. (30 points) Given a sequence x_1, x_2, $\cdots$, x_n of real numbers (not necessarily positive), a maximum subsequence x_i, x_{i+1}, $\cdots$, x_j is a subsequence of consecutive elements from the given sequence such that the sum of the numbers in the subsequence is maximum over all subsequences of consecutive elements. Below is a program that determines the sum of such a sequence.

```
Global_Max := 0;
Suffix_Max := 0;
i := 1;
while (i<=n) do
   if x[i] + Suffix_Max > Global_Max then
      Suffix_Max := Suffix_Max + x[i];
      Global_Max := Suffix_Max
   else
     if x[i] + Suffix_Max > 0 then
       Suffix_Max := Suffix_Max + x[i]
     else Suffix_Max := 0
     fi
   fi;
   i := i + 1
od;
```

Annotate the program into a standard proof outline, showing clearly the partial correctness of the program.

Solution. Let $isMS(s, x, i)$ denote that s is the sum of the maximum subsequence in $x[1..i]$ and $isMSX(s, x, i)$ denote that s is the sum of the maximum subsequence that is also a suffix in $x[1..i]$. In particular, $isMS(0, x, 0)$ and $isMSX(0, x, 0)$ both hold, as $x[1..0]$ denotes the empty sequence. To shorten formulae, we denote `Global_Max` and `Suffix_Max` respectively by G_M and S_M in all assertions.

```
// assume n ≥ 1, which is preserved by the code and will be omitted later
Global_Max := 0;
// isMS(G_M, x, 0)
Suffix_Max := 0;
// isMS(G_M, x, 0) ∧ isMSX(S_M, x, 0)
i := 1;
// inv: (1 ≤ i ≤ n + 1) ∧ isMS(G_M, x, i − 1) ∧ isMSX(S_M, x, i − 1)
while i <= n do
    // (1 ≤ i ≤ n) ∧ isMS(G_M, x, i − 1) ∧ isMSX(S_M, x, i − 1)
    if x[i] + Suffix_Max > Global_Max then
        // (1 ≤ i ≤ n) ∧ isMS(x[i] + S_M, x, i) ∧ isMSX(x[i] + S_M, x, i)
        Suffix_Max := Suffix_Max + x[i];
        // (1 ≤ i ≤ n) ∧ isMS(S_M, x, i) ∧ isMSX(S_M, x, i)
        Global_Max := Suffix_Max
    else
          // (1 ≤ i ≤ n) ∧ isMS(G_M, x, i) ∧ isMSX(S_M, x, i − 1)
          if x[i] + Suffix_Max > 0 then
              // (1 ≤ i ≤ n) ∧ isMS(G_M, x, i) ∧ isMSX(x[i] + S_M, x, i)
              Suffix_Max := Suffix_Max + x[i]
          else
              // (1 ≤ i ≤ n) ∧ isMS(G_M, x, i) ∧ isMSX(0, x, i)
              Suffix_Max := 0;
          fi
    fi;
    // (1 ≤ i ≤ n) ∧ isMS(G_M, x, i) ∧ isMSX(S_M, x, i)
    i := i + 1
od;
// isMS(G_M, x, i−1) ∧ isMSX(S_M, x, i−1) ∧ i = n+1(implying isMS(G_M, x, n))
```

□