Thinking Programs: Exercises

Chapter 8: Computer Programs

1. We consider the following two problems:

   a) Given an array a and a positive integer n, find the minimum of the first n elements of a, i.e., that element m that occurs at some of the first n positions of a and that is less than or equal all elements at these positions.

   b) Given an array a and a positive integer n, find an index of the minimum of the first n elements of a, i.e., a non-negative integer p less than n such that the element of a at p is that minimum.

   Formalize the specification of each problem. For this, do not use the arithmetic quantifier min but only the predicate logic quantifiers $\forall$ and $\exists$ (translate above specification from natural language to logic).

   For each problem specification, answer the following questions (with suitable justifications):

   - Is the precondition satisfiable? Is it not trivial?
   - For every input that satisfies the precondition, is the postcondition satisfiable? Is it not trivial?
   - Is for every input that satisfies the precondition the output uniquely defined by the postcondition?

2. Derive the weakest precondition of the command C defined as

   ```
   if (i < 10) { a[i] = a[i]+2; i = i-1; } i := i+1;
   ```

   for postcondition F defined as $a[i] = 7$ (ignoring "index out ouf bound" violations). Simplify the derived precondition as far as possible.

   Also derive for command c and precondition F the strongest postcondition. Simplify the derived postcondition as far as possible.

   Also derive for above command a judgement of form $c : [F]^{x,\dots}$ for some state relation F and variable frame $\{x, \dots\}$.

   Remember (for all parts) that an array assignment $a[i] := b$ is just an abbreviation for the scalar assignment $a := a[i \mapsto b]$.

3. Repeat Exercise 2 for the following command C

   ```
   i := i+1; if (i < 10) { a[i] = a[i]+2; i = i-1; }
   ```

   and condition F defined as $a[i] = 7$.

4. Repeat Exercise 2 for the following command C

```
if (i < 10) { a[i] = a[i]+2; i = i+1; } i := i+1;
```

and condition F defined as $a[i] = 7$.

5. Repeat Exercise 2 for the following command C

   ```
   i := i+1; if (i < 10) { a[i] = a[i]+2; i = i+1; }
   ```

   and condition F defined as $a[i] = 7$.

6. Take the following program which computes for inputs $m, n \in \mathbb{N}$ with $n \neq 0$ the truncated quotient $q := \lfloor m/n \rfloor$ and remainder $r := m - n \cdot q$:

   $\{m = oldm \land n = oldn \land n \neq 0\}$

   ```
   q = 0;
   r = m;
   while (r >= n)
   {
     q = q+1;
     r = r-n;
   }
   ```

   $\{oldm = n \cdot q + r \land r < oldn\}$

   E.g., for $a = 50$ and $n = 11$ we have finally $q = 4$ and $r = 6$.

   Assume you are given a suitable loop invariant I and termination measure T; using I and T state all verification conditions (classical logic formulas) that have to be proved for verifying the total correctness of the program (writing $I[t/x]$ for a substitution of term t for variable x in I and analogously for T).

   Construct for above inputs a table for the values of the variables before/after each loop iteration. Using this table as a hint, give suitable definitions for I and T. Demonstrate how from your choice of I and T the verification condition can be proved.

7. Take the following program which computes for given $n \in \mathbb{N}$ the result $s := n^2$:

   $\{n = oldn\}$

   ```
   s = 0; i = 1;
   while (i <= n)
   {
     s = s+2*i-1;
     i = i+1;
   }
   ```

   $\{s = n^2 \land n = oldn\}$

Assume you are given a suitable loop invariant I and termination measure T; using I and T state all verification conditions (classical logic formulas) that have to be proved for verifying partial correctness and termination of the program (writing $I[t/x]$ for a substitution of measure t for variable x in I and analogously $T[t/x]$).

Construct for input $n = 10$ a table for the values of the variables before/after each loop iteration. Using this table as a hint, give suitable definitions for I and T. Demonstrate how from your choice of I and T the verification condition can be proved.

8. Take the following program which computes for input array a and natural number n the output $x := \sum_{i=0}^{n} a[i] \cdot 10^i$ (i.e., it computes the natural number x denoted by the $n+1$ decimal digits in array a):

   $\{a = olda \wedge n = oldn\}$

   ```
   x = 0;
   while (n >= 0)
   {
     x = 10*x+a[n];
     n = n-1;
   }
   ```

   $\{x = \sum_{i=0}^{oldn} olda[i] \cdot 10^i\}$

   E.g., for $a = [3,1,4,7]$ and $n = 3$ we have finally $x = 7413$.

   Assume you are given a suitable loop invariant I and termination measure T; using I and T state all verification conditions (classical logic formulas) that have to be proved for verifying partial correctness and termination of the program (writing $I[t/x]$ for a substitution of term t for variable x in I and analogously $T[t/x]$).

   Construct for above inputs a table for the values of the variables before/after each loop iteration. Using this table as a hint, give suitable definitions for I and T. Demonstrate how from your choice of I and T the verification condition can be proved.

9. Take the following piece of code which computes for inputs $x, y \in \mathbb{N}$ the output $p := x \cdot y$:

   $\{x = oldx \wedge y = oldy\}$

   ```
   p = 0;
   while (y > 0)
   {
     if (y % 2 == 0)
     {
       x = x*2; y = y/2;  // y is even
     }
     else
     {
       p = p+x; y = y-1;  // y is odd
     }
   ```

```
  }
```

$\{p = oldx \cdot oldy\}$

Assume you are given a suitable loop invariant I and termination measure T; using I and T state all verification conditions (classical logic formulas) that have to be proved for verifying partial correctness and termination of the program (writing $I[t/x]$ for a substitution of term t for variable x in I and analogously $T[t/x]$).

Construct for inputs $x = 5, y = 10$ a table for the values of the variables before/after each loop iteration. Using this table as a hint, give suitable definitions for I and T. Demonstrate how from your choice of I and T the verification condition can be proved.

10. Take the following program which computes for $a \in \mathbb{N}$ and $n \in \mathbb{N}$ the result $b = a^n$:

    $\{a = olda \land n = oldn\}$

```
  b = 1;
  while (n > 0)
  {
    if (n % 2 == 0) // n is even (i.e. 2|n)
      n = n/2;
    else
    {
      n = (n-1)/2;  // n is odd, thus n-1 is even
      b = b*a;
    }
    a = a*a;
  }
```

    $\{b = olda^{oldn}\}$

    E.g., for $a = 10$ and $n = 25$ we have finally $b = 10^{25}$.

    Assume you are given a suitable loop invariant I and termination measure T; using I and T state all verification conditions (classical logic formulas) that have to be proved for verifying partial correctness and termination of the program (writing $I[t/x]$ for a substitution of term t for variable x in I and analogously $T[t/x]$).

    Construct for above inputs a table for the values of the variables before/after each loop iteration. Using this table as a hint, give suitable definitions for I and T. Demonstrate how from your choice of I and T the verification condition can be proved.

11. Consider problem (a) from Exercise 1. We claim that this problem is solved by the following algorithm:

```
int m = a[0];
int i = 1;
while (i < n)
{
  if (a[i] < m) m = a[i];
```

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

Assume you are given a suitable loop invariant I and termination measure T; using I and T state all verification conditions (classical logic formulas) that have to be proved for verifying partial correctness and termination of the program (writing $I[t/x]$ for a substitution of term t for variable x in I and analogously $T[t/x]$).

Construct for some inputs a table for the values of the variables before/after each loop iteration. Using this table as a hint, give suitable definitions for I and T. Demonstrate how from your choice of I and T the verification condition can be proved.

12. Consider problem (b) from Exercise 1. We claim that this problem is solved by the following algorithm:

```
int m = a[0];
int p = 0;
int i = 1;
while (i < n)
{
  if (a[i] < m) { m = a[p]; p = i; }
  i = i+1;
}
```

Assume you are given a suitable loop invariant I and termination measure T; using I and T state all verification conditions (classical logic formulas) that have to be proved for verifying partial correctness and termination of the program (writing $I[t/x]$ for a substitution of term t for variable x in I and analogously $T[t/x]$).

Construct for some inputs a table for the values of the variables before/after each loop iteration. Using this table as a hint, give suitable definitions for I and T. Demonstrate how from your choice of I and T the verification condition can be proved.

13. Consider the problem of replacing in the first n positions of a every occurrence of an element x by an element y (leaving all other elements unchanged). Furthermore, variable r is to be set to the smallest position where a replacement has been performed ($r = -1$, if no replacement has been performed).

First, formally specify this problem by giving a suitable precondition and postcondition.

We claim that this problem is solved by the following algorithm:

```
int r = -1;
int i = n-1;
while (i >= 0)
{
  if (a[i] == x)
  {
    a[i] = y;
    r = i;
```

```
    }
    i = i-1;
  }
```

Assume you are given a suitable loop invariant I and termination measure T; using I and T state all verification conditions (classical logic formulas) that have to be proved for verifying partial correctness and termination of the program (writing $I[t/x]$ for a substitution of term t for variable x in I and analogously $T[t/x]$).

Construct for some inputs a table for the values of the variables before/after each loop iteration. Using this table as a hint, give suitable definitions for I and T. Demonstrate how from your choice of I and T the verification condition can be proved.

14. We are given an integer array a and an integer x that might appear as an element in a. Furthermore, we are given two integers *from* and *to* such that the closed interval [*from*, *to*] describes a range of indices in a. We assume that a is sorted in ascending order within this range (the order is not strictly ascending, i.e., the array may hold multiple identical elements). Our goal is to find an integer r that is either -1 or an array index in the given range. If r is -1, then x does not occur as an element of a in this range; otherwise, r is an index in this range at which a holds x. For instance, for inputs $a = [2, \underline{3, 3, 5, 7}, 11, 13]$, *from* $= 1$, *to* $= 4$, and $x = 5$, we expect output $r = 3$. For the same inputs except for $x = 11$, we expect output $r = -1$.

    First, formally specify this problem by giving a suitable precondition and postcondition.

    We claim that this problem is solved by the following Java code fragment that implements the core of the "binary search" algorithm:

    ```
    int r = -1; int low = from; int high = to;
    while (r = -1 && low <= high)
    {
      int mid = (low+high)/2;
      if (a[mid] == x)
        r = mid;
      else if (a[mid] < x)
        low = mid+1;
      else
        high = mid-1;
    }
    ```

    Please note that here `a/b` here means $\lfloor a/b \rfloor$.

    Assume you are given a suitable loop invariant I and termination measure T; using I and T state all verification conditions (classical logic formulas) that have to be proved for verifying partial correctness and termination of the program (writing $I[t/x]$ for a substitution of term t for variable x in I and analogously $T[t/x]$).

    Construct for above inputs a table for the values of the variables before/after each loop iteration. Using this table as a hint, give suitable definitions for I and T. Demonstrate how from your choice of I and T the verification condition can be proved.