Exercise Sheet 1

1. Consider the function REMOVE(v, x, n) which deletes all values greater than v from the array x of integers. The final result will be in $x[1], \ldots, x[t]$ and the function returns t.

```
REMOVE(v, x, n)
1.  t ← 0
2.  p ← 1;
3.  while (p ≤ n) do
4.      if x[p] ≤ v then
5.          t ← t + 1;
6.          x[t] ← x[p];
7.      p ← p + 1;
8.  return t;
```

a) Let $a_1, a_2, \ldots, a_n$ be the values of $x[1], \ldots, x[n]$ at line 1. Find a loop invariant that can be used to conclude that the function is correct (refer to line 3 to state the invariant).

b) Using the invariant, justify that the loop terminates.

c) Using the invariant, conclude that the value returned on line 8 satisfies the problem specification.

d) Prove the invariant, by mathematical induction.

e) Find the number of times each instruction is executed in the worst case and in the best case (define them as functions of n). For lines 3 and 4, count the number of times the condition is checked. Which properties characterize the best case and worst case instances of length n?

2. Consider the function FUNC(x, n, v, p), where x and v are arrays of integers of length n and p, and $1 \leq x[k] \leq p$, for $1 \leq k \leq n$.

```
FUNC(x, n, v, p)
1   for k = 1 to p do v[k] = 0
2   s = x[1]
3   v[s] = 1
4   t = 1
5   for k = 2 to n do
6       v[x[k]] = v[x[k]] + 1
7       if v[x[k]] > v[s] then
8           s = x[k]
9           t = 1
10      else if v[x[k]] = v[s] ∧ s ≠ x[k] then t = t + 1
11  return (s, t)
```

a) State the problem that FUNC(x, n, v, p) solves. Which is the meaning of s and t?

b) State an invariant of the loop 5-10 that allows us to conclude that the function is correct. Explain how the invariant is kept in each iteration and how we conclude that the result (s, t) in line 11 is correct.

c) Characterize the time complexity of FUNC(x, n, v, p), in terms of n and p. Explain your answer.

3. Consider a function that reads a sequence of integers that ends with `0` and outputs a tuple (v, k), where v is the **the last negative odd number** that was given and k is the **number of odd numbers** in the sequence. If there is no negative odd number, the value of v is `0`. Each line contains a single integer.

a) Write the function in pseudocode (let $x\%y$ be the remainder of the division of x by y). Do not use arrays.

b) Prove that the function you wrote is correct. For the proof, define a (useful) loop invariant and prove it.

c) Find the number of times each instruction is executed in the worst case and in the best case (define them as functions of the number n of input values). Characterize the best case and worst case instances.

4. *Binary Search.* Assuming you have an array v with n sorted in increasing order, we can use **binary search** to find if a given element x occurs in the array. Let m be the middle position: if $x = v[m]$ then we found it; if $x < v[m]$ then x can only be in the $v[1..m-1]$ half; if $x > v[m]$ then x can only be in the $v[m+1...n]$ half. This effectively reduces the search space by a factor of two.

a) Write pseudocode for an **iterative version** of this algorithm and prove its correctness using **loop invariants**.

b) Write pseudocode for an **recursive version** of this algorithm and prove its correctness using **recursion invariants**.

c) For the iterative version, find (almost) sharp bounds for the number of iterations of the loop, in the best case and in the worst case.

d) For the recursive version, find (almost) sharp bounds for the number of function calls, in the best case and in the worst case.

5. Consider the function REPETITIONS that finds all values that repeat consecutively in an array v, with n elements, and indicates them in $r[\,]$ in the same order they occur in v. In $r[\,]$, the value is followed by the number of times it repeats consecutively in v. Repetitions of the same value in different blocks are treated independently. The function returns the number of values repeated. Assume the indexes start at 1.

```
REPETITIONS(v, n, r)
1 | k ← 1;  j ← 0;
2 | while (k < n) do
3 |     c ← 0;   k ← k + 1;
4 |     while (k ≤ n ∧ v[k] = v[k − 1]) do
5 |         c ← c + 1;  k ← k + 1;
6 |     if c ≥ 1 then
7 |         r[j + 1] ← v[k − 1];   r[j + 2] ← c;
8 |         j ← j + 2;
9 | return j/2;
```

a) Which are the values of c, k, j and the state of the arrays when the instruction in line 6 is executed for the fifth time $v =$ `[2,2,2,3,4,3,3,6,6,6,6,2,2,5,6]` and $n = 15$. If it is unknown, write `??`.

b) Let $[a_1, a_2, \ldots, a_n]$ be the state of $v[\,]$ in line 1. Charaterize the sate of all variables when the condition in **line 2** is checked for the ith time. Charaterize the state for the tth test of the condition in **line 4** in iteration i of the block (for a fixed value of i). How can we use these conditions to conclude that the program is correct?