Homework Assignment 1

Due: Friday Jan. 21, 2022 (11:59pm)

(Total 40 marks)

CMPUT 204
Department of Computing Science
University of Alberta

Problem 1. (10 marks)

Consider the following definition of Fibonacci numbers:

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n \geq 2 \end{cases}$$

a. (3 marks) Give the set of all natural numbers q for which $F(q) \geq 1.6^{q-2}$. Justify briefly. (In the textbook, the set of natural numbers, denoted $\mathbb{N}$, starts from 0. This is what we adopt in this course.)

b. (3 marks) Does your answer to **a** imply the following assertion: $\exists c, n_0 > 0$, such that $0 \leq c \cdot 1.6^n \leq F(n)$, $\forall n \geq n_0$, where c is a real number and n_0 a natural number. Explain briefly.

c. (4 marks) Consider the following recursive implementation of function $F(n)$ (which differs slightly from the one given in the lecture notes).

```
procedure fib(n)
if (n = 0) then
     return 0
else if (n = 1 or n = 2) then
     return 1
else
     return fib(n − 1) + fib(n − 2)
```

Prove by induction that the procedure `fib`(n) is correct.

Problem 2. (10 marks) Consider the following recursive version of `InsertionSort`:

```
procedure InsertionSort(A, n)
    **Sorts array A of size n
    if n > 1 then
        InsertionSort(A, n − 1)
        x ← A[n]
        PutInPlace(A, n − 1, x)
    end if

procedure PutInPlace(A, j, x)
    if (j = 0) then
        A[1] ← x
    else if x > A[j] then
        A[j + 1] ← x
    else            ** i.e., x ≤ A[j]
        A[j + 1] ← A[j]
        PutInPlace(A, j − 1, x)
    end if
```

a. (6 marks)

- [3 marks] First, prove (using induction) the correctness of `PutInPlace` by showing that:

  > For any array A, and natural number j such that (i) A has (at least) $j + 1$ cells, and (ii) the subarray $A[1..j]$ is sorted, when `PutInPlace(`A, j, x`)` terminates, the first $j + 1$ cells of A contain all the elements that were originally in $A[1..j]$ plus x in sorted order.

  We provide a partial solution here.

  We prove the predicate by induction. The base case is when $j = 0$, for which the claim is true as `PutInPlace` just puts x in the first cell of A.

  Induction step: Let $j \geq 1$ be an arbitrary integer and assuming the claim holds for $j - 1$ we show it also holds for j. If x is greater than $A[j]$, then x is greater than all elements in $A[1..j]$, so by putting x in the $j + 1$-th cell the array $A[1..j + 1]$ satisfies the required: sorted, and contain all the required elements. ... **You are asked to complete this proof.**

- [3 marks] Use induction and the correctness of `PutInPlace()` to prove the correctness of `InsertionSort(`A, n`)`.

b. (4 marks) For each state of array A below, indicate whether it ever occurs during the call, `InsertionSort(`A, n`)`, where $A = [4, 7, 2, 8, 6, 5]$ and $n = 6$.

i. $A = [4, 2, 7, 8, 6, 5]$

ii. $A = [2, 4, 7, 8, 6, 5]$

iii. $A = [2, 4, 7, 6, 8, 5]$

iv. $A = [2, 4, 6, 7, 8, 5]$

v. $A = [2, 4, 6, 7, 5, 8]$

vi. $A = [2, 4, 6, 5, 7, 8]$

vii. $A = [2, 4, 5, 6, 7, 8]$

Problem 3. (8 marks) Consider the following code, which takes as input an array A of n integers, where $n > 0$.

```
procedure Unknown(A, n)
i ← 1
p ← A[1] + 1
while (i < n) do
    j ← i + 1
    p ← p + (A[j] + j)
    i ← i + 1
return p
```

(i) Suppose the input array is $A = [2, 5, 3, 6]$. What does the procedure return?

(ii) Give a loop invariant (LI) for the loop in the code.

Hint: Here is a partial analysis: At the start of each iteration i ($1 \leq i \leq n$), we have

- when $i = 1$, the value of p is $A[1] + 1$;
- when $i = 2$, the value of p is $(A[1] + 1) + (A[2] + 2)$;
- when $i = 3$, the value of p is $(A[1] + 1) + (A[2] + 2) + (A[3] + 3)$;
- .....
- when $i = n$, the loop terminates.

(iii) Prove the the properties of Initialization, Maintenance, and Termination (including both Termination #1 and Termination #2). Since the problem does not specify what procedure Unknown(.) is supposed to compute, in proving Termination #2 you only need to indicate what is returned by the procedure when the loop terminates and and argue why it is implied by LI.

Problem 4. (12 marks) Describe an algorithm for finding both the minimum and maximum of n numbers using fewer than $3n/2$ key comparisons.

Hints:

- The case of $n = 1$ or 2 is trivial, at most 1 key comparision is needed. When $n > 2$, one can use a loop to examine each element after the first two and conduct two key comparisons to determine whether it is a new candidate minimum/maximum. This algorithm requires $1 + 2(n - 2)$ key comparisons, which does not satisfy the condition given in the problem.
- To reduce key comparisons, process each pair of elements (after the first two) using a loop and determine if the smaller of the pair is a new candidate minimum and if the larger one is a new candidate maximum. You need to handle both cases when n is even or odd.

Here are what you need to do for this problem.

- Write a pseudocode for your algorithm, which must use exactly one loop (as sketched above, you don't need multiple loops).
- Explain why your algorithm uses fewer than $3n/2$ key comparisons.
- Write a loop invariant that your algorithm maintains and show the properties of Initialization, Maintenance, and Termination (including both Termination #1 and Termination #2)