Assignment 2

COMP206 (T4 2020)

Upload a single PDF named username_a2.pdf with all the answers. All questions carry 10 marks, except Q4 which is worth 20 marks. The last question (Q8) is a bonus question, it does not carry any marks. The assignment is for a total of 80 marks. The submission deadline is 11:59 pm, 16-Oct-2020.

1. Consider an integer data-type named `int4` which uses 4 bit twos complement for representing its integers.
    - (a) What is the number of distinct integers that can be represented using 4 bits?
    - (b) What is the minimum integer possible in `int4` ?
    - (c) What is the maximum integer possible in `int4` ?
    - (d) What are the bits in the internal representation of `4` and `-4` which are of the type `int4` ? [Hint: write all 4 bit binary numbers in a circle]
    - (e) Calculate the value of `z` by using twos complement in the following code:

```
int4 x = 4;
int4 y = -4;
int4 z = x+y;
```

2. Recall that for two integers `lo` and `hi` such that `lo <= hi` , the interval `{lo, lo+1, ..., hi-1}` is represented using `[lo, hi)` . When `lo=hi` , `[lo, hi)` represents an empty interval. Using this information solve the following:
    - (a) Enumerate elements in `[4, 7)` .
    - (b) Enumerate the elements in `[7, 9)` .
    - (c) Verify that `[4, 7)` ∪ `[7,9)` = `[4,9)` .
    - (d) Count the number of elements in `[lo, hi)` , given that `lo <= hi` . Justify your answer.
    - (e) Find all the values of `x` for which both `[4, x)` and `[x, 9)` are non-empty intervals.

3. Assuming that each statement is counted as one step, what is the number of steps in the execution of the following code in terms of `n` ? What is the runtime complexity in big-$\mathcal{O}$ notation?

```
int f(int n)
//@requires n>0;
{
    int total = 0;
    for(int i=0; i<n; i++)
    {
        for(int j=0; j<i; j++)
        {
            total = total + i + j;
        }
    }
    return 0;
}
```

4. We want to find the runtime complexity of the function `sort` given below. It utlizes a `swap` function, which on a function call of `swap(A, i, j)` exchanges the values of `A[i]` and `A[j]`. To find the runtime complexity we need to find inputs for which it takes the maximum amount of time (worst case running time). Trace the execution of the program to find how many times the `swap` function is called when the input array `A` is:

    - (a) A = [0,1,2]
    - (b) A = [1,2,3,0]
    - (c) A = [3,2,1,0]

    To trace the execution of the inputs above, track the changes in the variables by creating a table. Fill in all possible values taken by variables i and j during the execution of the algorithm, before the inner loop guard is tested.

    $n = 3, A = [0, 1, 2]$

i	j	swap called?	A
1	1	no	[0,1,2]
2	2	..	..
..	..	..	..

```
int sort(int[] A, int n)
//@requires n == \length(A);
{
    int i = 1;
    //outer loop
    while (i < n)
    {
        int j = i;
        //inner loop
        while (j != 0 && A[j-1] > A[j])
        {
            swap(A, j-1, j);
            j = j - 1;
        }
        i = i + 1;
    }
}
```

Based on your observations above, what is the property satisfied by arrays which make the code above run longest? If such a worst case array has `n` elements how many times is the `swap` function called? What is the runtime in big-$\mathcal{O}$ notation?

5.
   - (a) Rewrite the `sort` function in the question above by changing all the while loops to for loops.
   - (b) When j=0, the inner loop guard in the `sort` function reads as `0!=0 && A[-1] > A[0]`. Note that `A[-1]` violates the safety requirement for array indices. Is this a bug? If not, why?

6. Write contracts for the following specifications:
   - the function signature is `int xyz(int[] A, int lo, int hi)` : The inputs `lo` and `hi` refer to some non-empty interval used for indexing `A[lo, hi)`, and the function returns an index in the interval `[lo, hi)`.
   - the function signature is `int gcd(int a, int b)` : Both the inputs are non-zero and positive. The returned value divides both `a` and `b`, and the function does not modify `a` and `b` anywhere in its body.

7. Prove the loop invariant and the assertion. Also prove that the loop terminates.

```
void scanInterval(int lo, int hi)
//@requires 0 <= lo && lo <= hi;
{
    int i;
    for ( i = lo; i < hi; i++ )
    //@loop_invariant lo <= i && i <= hi;
    {
        //body
    }
    //@assert i==hi;
}
```

8. [BONUS QUESTION, THIS CARRIES ZERO MARKS] Consider a two player game in which there is a heap of `3n` coins, for some natural number `n`. The game proceeds with each player taking turns to remove either one coin or exactly two coins. The player who removes the last coin wins the game. Suppose you and Alice are playing this game and you start the game by picking some coins. Alice follows a strategy of picking `3-x` coins if `x` coins were picked by you in the previous move.

    - (a) What is the number of coins required on the heap so that you can win the game in one move?
    - (b) Starting with 6 coins, show that no matter what your moves are you can not win the game.
    - (c) One can model the game play as a loop, (1) you make a move and then, (2) alice makes a move and then (1) and (2) are repeated till we have a winner. Show the loop invariant that the number of coins in the heap before you make a move is always a multiple of 3. Use the loop invariant to prove that for any game which starts with `3n` coins, Alice will always win the game no matter how smartly you play.
    - (d) As Alice can always win when the number of coins is a multiple of 3, we say that Alice has a winning strategy in this case. Who has the winning strategy when the number of coins is not a multiple of 3? Why?