STUDENT NAME: _____

STUDENT ID: _____

McGill University
Faculty of Science
School of Computer Science

MIDTERM EXAMINATION

COMP-250: Introduction to Computer Science

March 12, 2014

Examiner: Prof. Doina Precup

Write your name at the top of this page. Answer directly on exam paper (use both sides of the paper). There are 3 questions worth 100 points in total. Partial credit will be given for incomplete or partially correct answers.

SUGGESTIONS: READ ALL THE QUESTIONS BEFORE YOU START!
GOOD LUCK!

1. [40 points] **An array problem**

   You are given an array a of integers of size n, whose elements are all distinct numbers (no two elements are equal) and a value x.

   (a) [30 points] Write an algorithm CountPairs (either in pseudocode or in Java) that returns the number of distinct pairs of elements in a have the sum x.

   For example, on the array: 1 4 3 2 5, calling with $x = 5$ would return 2, calling with $x = 1$ would return 0 and calling with $x = 3$ would return 1.

   You are allowed to rearrange the elements of the array. Make your algorithm as efficient as possible. You may call any of the algorithms we discussed in class (sorting, max, search etc), and you do NOT need to reproduce their code here.

   (b) [10 points] Give the $O()$ for the running time of your algorithm in terms of the size of the array n, and explain your answer in 2-3 sentences. No formal proof is necessary.

2. [20 points] **Short questions**

   (a) [5 points] Consider the following Java method:

```
public static int mystery(int n) {
int i = 1;
int s = 0;
for (int i = 1; i < n; i = i * 2)
   s++;
return s;
}
```

   What function of n does the code compute?

   (b) [5 points] The Stack data structure has three main operations: push, pop and top. As discussed in class, stacks can be implemented using either arrays or lists. Assuming in the array implementation, enough memory is allocated for the stack to not exceed capacity, is there any difference in terms of $O()$ for these operations in the two implementations?

(c) [10 points] Consider the following Java code:

```
public class Test{
  int n;

  public Test() {
    n = 0;
  }

  public void incr(int i) {
    i++;
    n = n + i;
  }

  public int get() {
    return n;
  }

  public static void main(String[] args) {
    Test x = new Test();
    int i = 3;
    x.incr(i);
    System.out.println("i="+i);
    System.out.println("n="+x.get());
  }
}
```

What will the code print? (no explanation necessary)

3. [40 points] **Recursion**

   You are given an array a of integers of size n.

   (a) [25 points] Write ***a recursive algorithm***, called IsSumConstant, which returns true if the sum of all pairs of elements whose indices sum to $n-1$ is the same, i.e. $a[0]+a[n-1] == a[1]+a[n-2] == a[2]+a[n-3] == ....$ If the array is of odd size, you will add the middle element to itself and compare that sum to the rest.

   For example, on the array: 1 3 4 2, the algorithm should return false (1+2 is not the same as 3+4).

   On the array: 5, the algorithm should return true (we only have one sum).

   On the array: 3 1 4 2 the algorithm should return true (because 3+2 is the same as 4+1).

   On the array: 1 2 3 4 5 the algorithm should return true (because 1+5, 2+4 and 3+3 are all the same).

   You are allowed to write more than one method as part of the algorithm. You may write either pseudocode or Java, as you prefer.

   (b) [10 points] Write a recurrence for the running time of your algorithm.

   (c) [5 points] Based on this recurrence, what is the $O()$ of your algorithm?