STUDENT NAME: ______________________________

STUDENT ID: ______________________________

McGill University
Faculty of Science
School of Computer Science

MIDTERM EXAMINATION

COMP-250: Introduction to Computer Science

October 20, 2011

Examiner: Prof. Doina Precup

Write your name at the top of this page. Answer directly on exam paper. Two blank pages are added at the end in case you need extra space. There are 4 questions worth 150 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. [50 points] **An array problem**

   You are given two arrays, a of size n and b of size 2.

   (a) [35 points] Give an algorithm that counts how many times the content in array b is present in array a. You can solve the problem in pseudocode or Java, whichever you prefer.

   E.g. If array a is: 10 1 2 3 1 4 5 1 2 8 and array b is: 1 2, your algorithm should return 2, because 1 2 appears twice in a. If b is: 10 9, you should return 0. If b is: 2 3, you should return 1.

   Solution: We will go sequentially over a and look for the occurrence of $b[0]$, then look for the next element being $b[1]$. The pseudocode is as follows:

```
Algorithm CountOccurrence ($a$, $n$, $b$)
Input: An array $a$ of size $n$ and an array $b$ of size 2
Output: The number of times $b$ occurs in $a$

int $m = 0$ //This variable will hold the result

// Using Java-like syntax and indexing
for (int $i = 0$; $i < n - 1$; $i = i + 1$)
  if ($a[i] == b[0]$ and $a[i + 1] == b[1]$) then $m = m + 1$
return $m$
```

   (b) [10 points] Give the $O()$ of your algorithm in terms of n, and explain your answer in 1-2 sentences. No formal proof is necessary.

   Solution: This is $O(n)$, because we have a for loop going $n - 1$ times, and the operations inside the loop are constant time, so they have $O(1)$.

   (c) [5 points] Suppose instead of size 2, b contains k elements. Explain how your algorithm would change, and how O() would change. You do NOT need to write the new algorithm, just explain it based on the algorithm you wrote for part a.

   Solution: The "for" loop would go to $n - k$ and the **if** statement would be replaced by a **for** loop over all the elements of b, checking them against a. We would use a boolean flag initialized before the loop to signal if all elements of b are present in a. If this flag is true, the counter would be incremented. The following pseudocode corresponds to this solution. The pseudocode was not required for the exam, I am including it here for clarity.

2. [15 points] **Big-oh**

   State the $O()$ of the following pieces of code, and briefly justify your answer.

   (a) for (int $i = n; i > n/2; i - -)$
   System.out.println(i);

   Solution: $O(n)$, because the loop executes $n/2$ times.

   (b) for (int $i = 1; i < n; i = 2 * i)$
   System.out.println(i);

   Solution: $O(\log n)$, because the sequence of is in the loop is: $1, 2, 2^2, 2^4, \ldots$. The loop will terminate when $i = 2^k > n$, at which point we will have $k \approx \log_2 n$), and k is the number of loop executions.

   (c) for (int $i = 1; i < n; i = 2 * i - 1)$
   System.out.println(i);

   Solution: This is an infinite loop, because i is always set to 1.

3. [35 points] **Reading code**

   The following piece of code is supposed to count the number of even elements in an array. The code contains some mistakes - please identify what they are.

```
public static class CountEven {

   private int[] a;

    public int count() {
      int count = a[0];
      for (i = 1; i < a.length; i++)
          if (a[i] / 2 == 0) count++;
      return count;
   }
}
```

   Solution:

   (a) Class cannot be static

   (b) There is no constructor to allocate and initialize a, nor any other methods to do this

   (c) i should be declared as int

   (d) count should be initialized to 0, not a[0]

   (e) The if condition should used % (the mod operator), not /

4. [50 points] **A recursive algorithm**

   Suppose you are given two arrays of bits (0s and 1s). The Hamming distance between two arrays of length n is equal to the number of bits in which the two arrays differ. For example, the arrays: 0 0 1 0 and 1 0 1 0 have Hamming distance 1. Note that the distance between two arrays of size n has to be between 0 and n.

   Write an algorithm that takes as input an array of bits a of size n and a number k. The algorithm should *count* all the bit sequences that are within Hamming distance k of the given array.

   For example, if k is 2 and a is $\{0, 0, 0, 0\}$ then your algorithm should return 6, because there are 6 sequence within Hamming distance 2 of a:
   1100 1010 1001 0110 0101 0011

   You can use either Java or pseudocode, though pseudocode is recommended in this case. You can add arguments to the input of your algorithm (aside from a, n and k), if you think that is necessary.

Page left intentionally blank

Page left intentionally blank