

# CENG 707 – Data Structures and Algorithms

## Midterm Exam #1 03/11/2011

**NAME SURNAME, ID:**

---

**Duration:** 70 minutes

**Exam:**

- This is a **closed book, closed notes** exam. The use of any reference material is strictly forbidden.

**About the exam questions:**

- The points assigned for each question are shown in parentheses next to the question number.

**This exam consists of 6 pages including this page. Check that you have them all.**

**GOOD LUCK!**

---

**Question 1**

**Question 2**

**Question 3**

**Question 4**

**Total:**

**1** (30 points)



Suppose that you need to write a program to generate a single permutation of the first  $N$  positive integers. For example if  $N$  is 6,  $[3,1,2,5,4,6]$  is a legal permutation, i.e., each integer less than or equal to six should appear exactly once in this sequence. Assume you have a random number generator, ***randomInteger(i,j)***, that can generate integers between  $i$  and  $j$ . Analyze the **average and worst case running times** of the following algorithms using Big-Oh, i.e.,  $O(\dots)$ , notation.

(a) 10pts

Fill an array  $a$  from  $a[0]$  to  $a[N-1]$  as follows: To fill  $a[i]$ , generate random numbers between  $1$  and  $N$ , until you get one number that is not in  $a[0]$ ,  $a[1]$ , .....  $a[i-1]$

(b) 10pts

Same as the algorithm in part (a), but keep an extra array called the *used* array. When a random number, *k*, is first put in the array, set *used[k] = true*. This means that when filling *a[i]* with a random number, you can test in one step whether that number is used, instead of looking at all *i-1* numbers before *a[i]*.

(c) 10pts

Fill the entire array initially using *a[i]=i+1*. Then in a for loop, swap each entry with a random entry.

```
for (i=0;i<N;i++)  
    swap (a[i], a[ randomInteger(0,i) ] );
```

2

(30 points)



Write a recursive function, ***count(a,x)*** to count how many times a given integer, ***x***, exists in an array, ***a***, of integers. A non-recursive implementation uses a variable to store the temporary “count” as you go over the elements of the array. Your recursive implementation ***must not*** use any additional variable to store the result.

Example usage:

```
int a[10] = {4,3,0,3,4,2,1,4,3,2};
```

```
printf(“%d”,count(a,4) );
```

should print “3” on the screen.

3

(30 points)



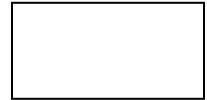
(a) 15pts

Given an example integer array of 10 elements in which quicksort performs in its worst case performance,  $O(n^2)$ . Assume that we always use the middle element as pivot during the partition operation. You do not need to show the operations. But briefly explain why it will be  $O(n^2)$ .

(b) 15pts

Describe an algorithm to sort an array of  $M+N$  integers in which the first  $M$  integers in the array are already sorted and the second part which contains  $N$  integers is not sorted. You may call existing sorting algorithms in your algorithm. What is the time complexity of your algorithm?

**4** (10 points)



Give an example of a “class template” and an example of a “function template”. You do not need to provide any implementation.