Makerere University
Faculty of Computing and IT
End Of Semester I Examination 2007/2008

Program: BSc. CS

Year of Study: II

Course Name: **Algorithms and Data Structures**
Course Code: **CSC 2100**

Date: 6 Dec. 2007
Time: 8:00am - 11:00am

Examination Instructions

- Attempt All questions in Section A (40 Marks). Each question in this section carries 4 marks.
- Attempt three (3) questions in Section B (60 Marks)
- Do not open this exam paper until told to do so.
- All rough work should be in your answer booklet
- Leave your answer book(s), scrap paper on the desk when you leave.

Section A

1. Show the resulting *stack* after the following operations on an initially empty stack

```
push(5)
push(6)
push(7)
pop()
push(8)
```

2. Show the resulting *queue* after the following operations on an initially empty queue.

```
put(5)
put(6)
put(7)
get()
put(8)
```

3. Show the *binary search tree* that results from inserting the following sequence of numbers into an intially empty tree. (You may assume that the left sub-tree of any node always has items less than or equal to the parent, while the right sub-tree has items greater than the parent.)

```
4, 5, 6, 1, 2, 2, 7
```

4. Show the binary tree that results from inserting the following sequence into an initially empty tree. (Assumptions as above.)

```
21, 19, 17, 13, 11, 9, 4, 2
```

5. Consider the following recursive procedure

```
int s(int a[], int item, int l, int r)
{
  int m = (l+r)/2;
  if (l>r) return a[l];
  else if (a[m] == item) return m;
  else if (a[m] > item) return s(a, l, m);
  else return s(a, m, r);
}
```

What is the result when s is called as $s(x, 14, 0, 5)$, where x is an array containing $2, 4, 5, 12, 14, 45$?

6. Explain what is meant by the "Order" of an algorithm (i.e. big-O notation).

7. Explain briefly how the *partition* method of Quicksort can be used to find the median of a set of items *without* first sorting the items.

8. Explain the definition of the set *NP* as it relates to computational complexity.

9. Briefly outline the key elements of the *Halting Problem* in Computability Theory.

10. Explain, with illustrations the purpose of the *Counter-clockwise Test* in Computational Geometry.

Section B

1. **a.** Briefly illustrate the workings of the *selection sort* algorithm. (5 marks)

   b. Briefly illustrate the workings of the *Quicksort algorithm* (5 marks)

   c. Using the last item as the pivot in Quicksort, explain with illustrations, why Quicksort performs very poorly (in terms of runtime) if the items are already in sorted order. Write down the expected order in this case (10 marks)

2. **a.** Consider the sequence of numbers $1, 4, 7, 22, 13, 19, 11, 1$. Draw the binary search tree that would result from inserting these items into an initially empty tree. (5 marks)

   b. Draw a *balanced* binary search tree containing the same elements. (5 marks)

   c. Explain the (mathematical) relationship between the *height* of a balanced binary search tree and the number of nodes. Illustrate your answer.(10 marks)

3. **a.** Briefly explain the use of the counter clockwise (CCW) test in determining whether two line segments intersect. (5 marks)

   b. In some line segment intersection scenarios, the two segments share a single common end-point (i.e. the segments "touch" at one end), explain how the CCW test(s) can be used to detect this scenario. (5 marks)

   c. When two line segments are co-linear, it is possible to determine whether they intersect ("overlap") or not using both CCW and other tests. Explain how CCW can detect this scenario, then explain (with simple illustrations) how to determine (beyond CCW) whether they intersect or not. (10 marks)

4. **a.** Briefly outline a simple algorithm that can be used to determine whether a given natural number N is prime or not. (5 marks)

   b. Breifly explain why, in checking whether a number N is prime, it is not necessary to check if it has divisors greater than $\sqrt{N}$. (5 marks)

   c. Explain the *sieve of eratosthenes* method for determining whether a number is prime or not. Explain why this method is faster/better than all other considered in class. (10 marks)

5. **a.** Consider the *Luhn Algorithm* for validating credit card numbers provided in your reading. Outline its workings. (5 marks)

   b. What is the running time of the algorithm as a function of the number length? Illustrate your answer. (5 marks)

c. Write a short C program that takes an array of numbers (digits) representing a credit card, and determines whether the number is valid or not. (10 marks)