THE UNIVERSITY OF TRINIDAD & TOBAGO

FINAL ASSESSMENT/EXAMINATIONS DECEMBER 2011

Course Code and Title: **DSAL2001 - DATA STRUCTURES AND ALGORITHMS**

Programme: **BACHELOR OF APPLIED SCIENCE IN COMPUTER ENGINEERING**

Date and Time: **11th, December 2012, 1:00pm – 4:00pm** Duration: 3 Hours

PLEASE READ ALL INSTRUCTIONS CAREFULLY BEFORE YOU BEGIN THIS EXAMINATION

Instructions to Candidates

1. **This paper has 3 pages and 3 sections.**
2. **You are required to answer all questions in Section A.**
3. **You are required to answer all questions in Section B.**
4. **You are required to answer all questions in Section C.**
5. **The question paper must be returned with the Answer script.**

Key Examination Protocol

1. Students please note that academic dishonesty (or cheating) includes but is not limited to plagiarism, collusion, falsification, replication, taking unauthorised notes or devices into an examination, obtaining an unauthorised copy of the examination paper, communicating or trying to communicate with another candidate during the examination, and being a party to impersonation in relation to an examination.
2. The above mentioned and any other actions which compromise the integrity of the academic evaluation process will be fully investigated and addressed in accordance with UTT's academic regulations.
3. Please be reminded that speaking without the Invigilator's permission is **NOT** allowed.

Section A

1. Given the following numbers :

13	28	14	16	12	8	20	4	6	9

   (a) Show the step by step configuration of the list after an Insertion Sort algorithm has been applied to the list. [5]

   (b) Write a Java method that accepts an array of numbers and performs the insertion sort algorithm on the argument list. [5]

2. Write a recursive method to find the highest number in an array. Use the following function header: **int highest (int[] a, int first, int last)** [5]

3. Write a Java program that reads a text file line by line. Each line in the file consists of a string that may or may not be a palindrome. Your program should use the ADT Stack to output "**YES**" if the word is a palindrome and "**NO**" otherwise. [10]

   E.g. Input: level

   Output: YES

   E.g. Input: colour

   Output: No

Section B

1. State briefly what is a binary tree and explain briefly how it differs from a Binary Search Tree. [5]
2. Given the following values: **60 67 23 45 15 20 2 93 62 51 39 64 24 47**
   (a) Construct a Binary Tree using root as **60**. [4]
   (b) Display the in-order traversal [2]
   (c) Display the post-order traversal [2]
   (d) Display the pre-order traversal [2]
   (e) When a node is deleted it is replaced with its in-order successor. Show the construction of the tree when **45** is deleted. [3]
3. Write a recursive Java method to insert a node into a Binary Search Tree. Use the following function header. [8]

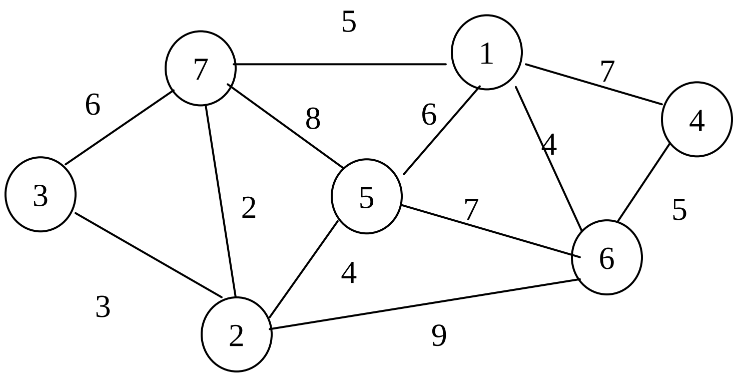
   void insertBST (Node root, Node node)

4. Write the recursive methods to traverse a BST using the following function headers:
   a. **void inOrder (Node root)** [3]
   b. **void preOrder (Node root)** [3]
   c. **void postOrder (Node root)** [3]

Section C

1. A hash table int [ ] list of size m, is used to store positive integers. A special key ***-1*** is used to represent an empty slot while ***-2*** is used to represented a deleted slot. Values are inserted into the key using double hashing with linear probing. The primary hash function is ***h_1 (k) = k* mod *m*** and the secondary hash function ***h_2(k) = 7 – k mod 7; where m = 11***
   (a) Write Java methods for the hash function h1 and h2. [4]
   (b) Show the hash table of size 11 after inserting entries with keys 34, 29, 53, 44, 120, 39, 45, and 40 using double hashing. [2]
   (c) Write an insert method using function header; **public void insert(int key, DataItem item)** to insert values into a hash table using Double Hashing. [4]

2. Given the following graph:



   (a) Draw the breadth-first traversal of the graph starting at node 1. Edges are processed in non-decreasing order. Clearly show your steps. [10]
   (b) Draw the minimum cost spanning tree using Kruskal's algorithm. At each stage show the connected sub-graphs represented as a collection of disjoint sets. [10]