THE UNIVERSITY OF THE WEST INDIES
ST. AUGUSTINE

EXAMINATIONS OF December 2022

Code and Name of Course: COMP 2611 – Data Structures

Date and Time: *Monday 19th December 2022* *1:00 pm* Duration: 2 Hours

INSTRUCTIONS TO CANDIDATES: This paper has 5 pages and 4 questions

The use of silent, non-programmable, scientific calculators is allowed.
Answer ALL questions.
Questions are NOT equally weighted.

1. Consider the following declarations of a node in a linked list and some typical stack operations:

```
struct LLNode {
   int data;
   LLNode * next;
};

Stack * initStack ();
bool isEmptyStack (Stack * s);
void push (Stack * s, LLNode * data);
LLNode * pop (Stack * s);
```

a) Write a recursive function, *isSortedRec*, which returns *true* if the elements of the linked list passed as a parameter are sorted in ascending order, and *false* otherwise. Its prototype is:

```
bool isSortedRec (LLNode * top);
```

[4 marks]

b) Write an iterative (i.e., non-recursive) function, *reverseList*, which reverses the elements of a linked list and returns the top of the reversed list. No nodes must be deleted, and no new nodes must be created. You must use a stack with the given declarations to reverse the elements. The prototype of the *reverseList* function is:

```
LLNode * reverseList (LLNode * top);
```

NB: The usual functions of a linked list *may* be used in answering this question.

[6 marks]

Total Marks 10

2. Consider the following declaration of a node in a binary tree:

```
struct BTNode {
      int data;
      BTNode * left;
      BTNode * right;
      BTNode * parent;
};
```

A binary search tree (BST) is said to be *degenerate* if all the nodes in the tree have at most one child. Assume that a **BST with ≤ 1** node is degenerate.

a) Draw the **two** BSTs resulting from insertion of the following keys (in the order given) and determine which of them are degenerate (if any):

(i)	20	70	100	90	80	85	81	83	
(ii)	70	50	35	40	20				[3 marks]

b) Write a recursive function, *isDegenerate*, with the following prototype, which returns *true* if the BST rooted at *root* is degenerate and *false*, otherwise.

```
bool isDegenerate (BTNode * root);
```

[5 marks]

c) A BST contains integer keys. For each of the following sequences, state whether it could be the sequence of values examined in searching for the number **36**. If it cannot, state why.

```
7  25 42 40 33 34 39 36
92 22 91 24 89 20 35 36
95 20 90 24 92 27 30 36
```

[4 marks]

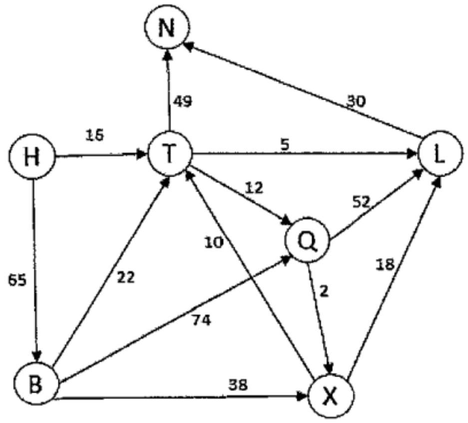
d) Write a function, *deleteMinBST*, which, given the root of a BST, deletes the node with the **smallest** key in the BST and returns the root of the modified BST. Your function must handle all cases (e.g., the node to be deleted is (i) the root, (ii) a leaf, or (iii) a non-terminal node). You can use any of the usual binary tree or BST functions **except** those that perform deletion. The prototype of the function is:

```
BTNode * deleteMinBST (BTNode * root);
```

[5 marks]

Total Marks 17

3. a) Given the following directed graph:



i) Draw the adjacency list representation of the graph. List the nodes in alphabetical order. [2 marks]

ii) Give the depth-first and breath-first traversals of the graph starting at ***B***. Edges leaving a node are processed in alphabetical order. [4 marks]

iii) Suppose that the graph is undirected. Draw the minimum spanning tree obtained by using Kruskal's algorithm. Show the steps in your derivation. [5 marks]

b) The following are some declarations for a directed graph represented as an adjacency list:

```
struct Edge {
   int dest;
   int weight;
   Edge * nextEdge;
};

struct Vertex {
   string ID;
   int colour;
   Edge * firstEdge;
};

struct Graph {
   int numVertices;
   Vertex vertices[100];
};
```

i) Write a function, *hasEdge*, with the following prototype which returns *true* if there is an edge between vertex *u* and vertex *v* in the graph *g*, and *false* otherwise:

```
bool hasEdge (Graph * g, int u, int v);
```
[3 marks]

ii) The *in-degree* of a vertex *v* in a graph *g* is the number of edges entering *v*. Using the *hasEdge* function from b) i) or otherwise, write a function, *inDegree*, with the following prototype which returns the in-degree of a vertex *v*:

```
int inDegree (Graph * g, int v);
```
[3 marks]

Total Marks 17

4. a) Consider the following declarations for a max-priority queue which is implemented using an underlying max-heap:

```
struct MaxHeap {
      int A [1000];
      int size;
};

struct MaxPriorityQueue {
      MaxHeap * heap;
};
```

A max-priority queue is to be populated from the elements of the following array, *A*:

10	60	5	25	70	65	45	50	15	80

Draw the max-priority queue, *maxPQ*, as a binary tree after **each** of the following operations takes place, **in the order given** (i.e., 5 binary trees must be drawn):

(i) `MaxPriorityQueue * maxPQ = initMaxPQ (A, 10);`
(ii) `int x = maximumPQ (maxPQ);`
(iii) `insertMaxPQ (maxPQ, 75);`
(iv) `x = extractMaximumPQ (maxPQ);`
(v) `increasePriority (maxPQ, 8, 90);`

[8 marks]

b) The array *A* has the following 8 elements:

55	10	70	90	30	40	20	50
0	1	2	3	4	5	6	7

The *partition* function from the quicksort algorithm is called as follows:

```
int q = partition (A, 0, 7);
```

Draw the array *A* after each pass of the *for* loop in the *partition* function and just before the function returns (8 diagrams in all). Assume that the pivot is *A*[7].

[8 marks]

Total marks 16

End of Question Paper (Total Marks 60)