National University of Computer and Emerging Sciences, Lahore Campus

Course:	**Data Structures**	**Course Code:**	**CS 201**
Program:	**BS(Computer Science)**	**Semester:**	**Fall 2017**
Duration:	**180 Minutes**	**Total Marks:**	**100**
Paper Date:	**14-Dec-2017**	**Page(s):**	**7**
Section:	**ALL**	**Section:**	
Exam:	**Final exam**	**Roll No:**	

Instruction/Notes:
All students can bring **one hand written A4 help sheet** in the exam hall
Sharing of A4 sheet is strictly **NOT ALLOWED**
Answer in the space provided
You can ask for rough sheets but **they will not be graded or marked**
In case of confusion or ambiguity make a reasonable assumption. Questions are not allowed

Good luck!

Question: 1 (Marks: 15)

Write a function ***Destroy()*** of the class AdjacencyList that de-allocates all the dynamic memory acquired by the Adjacency list. You can use the following definition of the AdjacencyList class

class AdjacecnyList
{ int v; // number of vertices in the graph
node **head;
};
//head is an array of pointers of size v. Each pointer marks the beginning of
// a list.
	class node
	{ int data;
	node *next;
	friend class AdjacecnyList;
	};

If you need to call another function from within ***Destroy()***, then also provide its definition.

Question 2 **(Marks: 15)**

You have a singly linked list called signal. Each node in the signal contains a real numbered value. Hence the node has the following structure:
The signal is stored in the non-decreasing order of values. We need to process this signal for possible truncation, i.e. reduction in size.

```
struct Node{
  double value;
  Node * next;
};
```

The criteria used for truncation is the following: any longest consecutive sequence of nodes whose values are less than 1.0 apart from the lowest value in the sequence are truncated to a single node containing the lowest value.

Following is an example of a linked list before and after processing. Here the letter X denotes the null pointer. As you can see, 1.4 is the smallest value in the sequence 1.4, 1.9 and 2.1, and all of these values have a difference of less than 1.0 from 1.4, therefore these are truncated to a single node containing 1.4 — and so on for the rest of the list.

Before
1.4 → 1.9 → 2.1 → 2.5 → 3.7 → 4.1 → 4.5 → 5.0 → 5.6 → X

After
1.4 → 2.5 → 3.7 → 5.0 → X

Write a C++ function called TruncateList which accepts the head pointer of a list and truncates the list according to the described criteria. Make sure there are no memory leaks in your program. If you use a helper function, then implement it too.

Question 3 (Marks: 15)

Implement a **recursive** function **CreateTree**, which is a member of the BST class to create a **balanced binary search tree** when given a sorted array in input. The function should return the root node of the newly created BST. You are **not allowed** to use any global or static variables. Also, you are **not allowed** to call any other helper function. The header of the function looks like the following:

```
class Node
{       int data;
        Node *right;
        Node *left;
        friend class BST;
};
```

Node *CreateTree(int *arr, int first, int last)

//first and last are the first and last indices of the array under consideration

For example, given the array: {2,3,4,9,21,25,26,27}, we will call the function as follows:

root = CreateTree(arr,0,7).

Roll Number:_____________ Section: __________

Question 4 (5+5+5+5+5+5+10+5+5+5)

a. Consider the following algorithm for finding the leaf with the minimum value in a max heap of size n. (n is an odd number and indexing starts from 1)

i) Start from i = root
ii) If the left child of i has smaller value than the right child, i=2i
else i=2i+1
iii) if i > n/2 STOP
else goto (ii)

Does this algorithm always return the leaf with the minimum value? If yes, give an argument for why that is the case. In no, show a binary heap for which the algorithm fails.

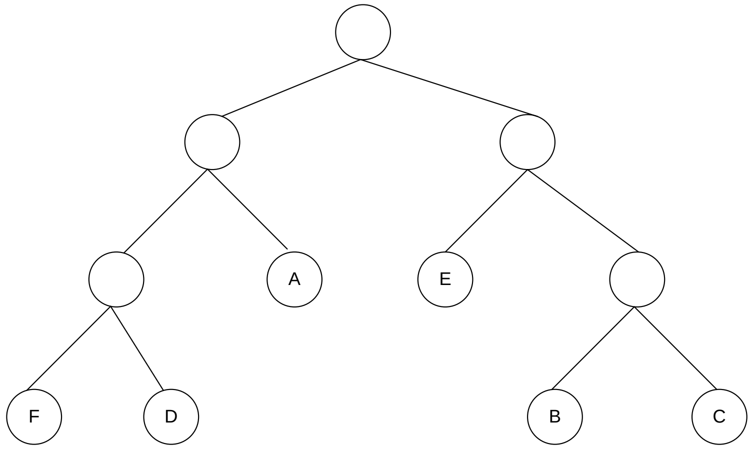
b. You wish to add a method called deleteAny(x) to your binary max heap class. This function will remove the key at index x in the heap. What is the fastest possible time in which you can accomplish this? The function must maintain both the structural and heap properties. Justify your answer with an example.

c. You are given a stack and asked to reverse its elements. You are given the choice to use either an extra stack or an extra queue for this purpose. Which choice will you take and why?

d. What are the minimum and maximum number of edges in a simple undirected graph G having |V| vertices.

e. Consider a binary heap having a total of m levels numbered from zero to (m-1). What is the minimum and maximum number of leaf nodes it can have?

f. Given the following binary tree (Huffman tree) used for encoding,



Decode the message: **01 10 001 01 01 000 10 111 110 01 10 001 000 01 111**

g. What is the order (big-oh) of the following operations, given each data structure has N values: Justify your answers in one line only.

i. InsertInOrder( ) into a sorted doubly linked list____

____

ii. Dequeue /Delete( ) from an array based queue____

____

iii. Search ( ) in a binary search tree____

____

iv. Push ( ) in a stack implemented using a singly linked list_______________________________________

____

v. Insert ( ) in max heap____

____

h. For the following grid apply **Depth First Search** and **specify the order of nodes visited when finding a path from the initial state A to the final state L.** You can move left, right, up, down. The cell named BLOCKED means you cannot go into that cell as it's a dead end. In case you have to choose between two cells, follow the cell whose name comes first in ascending order. For example if you have to choose between {B,E,G,J}, then choose B.

BLOCKED	J	K	L
E	F	G	BLOCKED
A	B	C	D

i. If you use two stacks to implement a queue and you can enqueue in O(1) then what is the complexity of dequeue? Justify with an example of queue (showing the contents of the stack) having the three items A,B,C.

j. Take an initially empty hash table of size 5, with hash function **h(x) = (x+1) mod 5**, and with collisions resolved by chaining. Draw the hash table after inserting the following sequence of keys into it: 35, 2, 28, 16, 13, 10, 18, 15