National University of Computer and Emerging Sciences, Lahore Campus

Course:	Data Structures	Course Code:	CS2001
Program:	BS (CS, SE, DS)	Semester:	Fall 2021
Duration:	180 Minutes	Total Marks:	60
Paper Date:	22-Jan-2022	Page(s):	16
Sections:	ALL	Section:	
Exam:	Final Exam	Roll No:	

Instruction/Notes: Answer in the space provided. You cannot ask for rough sheets since they are already attached. Rough sheets **will not be graded or marked.** In case of any confusion or ambiguity, make a reasonable assumption.

Question 1: Give answers to the following (Marks: 2*5 + 5*5)

1) How many leaf nodes are there in a full binary tree having 1023 nodes? A full binary tree is that in which every node other than the leaves has two children, and all leaves are at the same level. Show your complete working.

Leaf Nodes = 512

In a full binary tree total nodes are 2^k-1 if there are k levels.

If K = 10 then $2^{10}-1 = 1023$ total nodes

Last level number is K-1 starting root at 0 and

Number of nodes at last level = 2^{k-1}

Total nodes at last level $2^{10-1} = 512$

2) How many **minimum** numbers of nodes can be there in a **complete** binary tree having k levels? A complete binary tree is a **binary tree in which all the levels are completely filled except possibly the lowest one, which is filled from the left**. You can assume that root is at level zero. Show Complete Working.

$\mathbf{2^{k-1}}$

If K = 10 then $2^{10}-1 = 1023$ maximum nodes,

Tree is full till level k-1, total nodes till level k-1 = $2^9 -1= 511$

Add one left most node at last level k,

So minimum number of nodes in complete binary tree are $\mathbf{2^{k-1}} = 512$ that is equal to total number of leaf nodes in a full binary tree.

3) For which of the following does there exist a tree satisfying the specified constraint? Justify your Answer.
   a. A binary tree with 65 leaves and total 7 levels
   b. A binary tree with 33 leaves and total 6 levels
   c. A full binary tree with 64 total nodes and total 6 levels
   d. A binary tree with 2 leaves and height 100

A skewed binary tree with one child at all levels except one level.

4) How many linked lists are used to represent a graph with **n** nodes and **m** edges, when using an adjacency list representation?

n lists.
One list against each node of the graph.

5) A function is added to the Max-Heap class which updates the value of a given key in the heap. Resultantly, some of the Max-Heap properties may be violated. Redraw the Max-Heap after each of the following updates. You can only use the predesigned functions of Heap class to fix those violations, so that the Max-Heap properties remain intact. In each case start from the original heap and also Provide the function names.

```
        30
      /      \
   25          19
  /    \      /  \
 17     22   18   5
/  \    / \
12 13  6   18
```

25 to 5 (Bubble down)

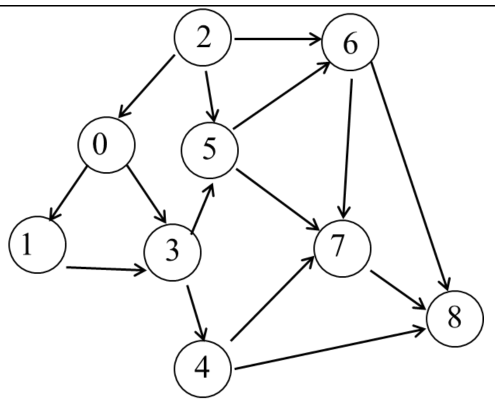
```
         30
       /     \
    22         19
   /    \     /  \
  17     18  18   5
 /  \    / \
12  13  6   5
```

```
        30
      /      \
   25          19
  /    \      /  \
 17     22   18   5
/  \    / \
12 13  6   18
```

12 to 29 (Bubble Up)

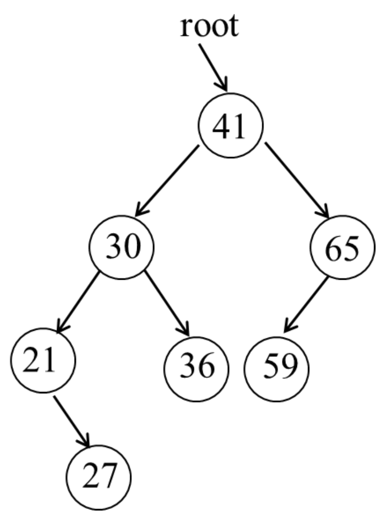
```
         30
       /     \
    29         19
   /    \     /  \
  25     22  18   5
 /  \    / \
17  13  6   18
```

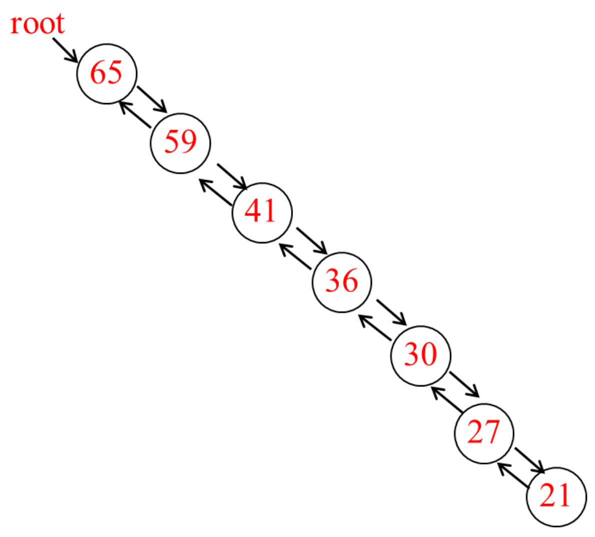
6) What are the outputs (order of visited nodes) of Depth First Traversal and Breadth First Traversal for the following graph, starting from node **0**, and breaking ties in the numerical order?

	**DFS** **0, 1, 3, 4, 7, 8, 5, 6**
	BFS **0, 1, 3, 4, 5, 7, 8, 6**

7) Redraw the following BST after execution of the code given below.

```
void UpdateTree(){
      if (root != nullptr) {
            Node* n1 = nullptr;
            UpdateTree(root, &n1);
            root = n1;
      }
}

void UpdateTree(Node* curr, Node** n1) {
      if (curr->left != nullptr)
            UpdateTree(curr->left, n1);

      curr->left = *n1;
      if (*n1 != nullptr)
            (*n1)->right = curr;

      *n1 = curr;
      if (curr->right != nullptr)
            UpdateTree(curr->right, n1);

}
```





8) Write true if the statement is true, else write false. Justify your answers. Credit will be awarded for correct justification only.

A. The maximum number of rotations required to balance an AVL tree after deletion is 2.

False

There could be more than one disbalanced cases need two rotations for LR and RL single case.

B. The hash code (hash function) of a key corresponds to an index number in the array.

True

C. If the Big-oh of insert operation in a hashmap (hash table) using probing is $O(1)$, then for the delete operation the Big-Oh would be $O(n)$.

False

Same probing will be used for deletion so cost will be O(1)

D. The worst-case complexity to find the minimum key in a max heap is $O(\log n)$.

False

Need to traverse complete heap for minimum value, so it will take O(n) time not O(log n)

E. A HashMap (hash table) cannot maintain the keys in sorted order.

True

The placement of Keys is random based on hash code.

9) Assume the singly linked list class **`SList`** has the head pointer of the list as its member, and **`SList`** is a friend of class `SNode` that is defined below.

```
class SNode {
    int element;
    SNode *next;
};
```

Write the C++ code of a member function **`deleteLastOccurrence`** for the class **`SList,`** which takes an integer element as parameter, and delete the last occurrence of this element from the list in $O(n)$ time. If the element does not exist in the list, then this function returns false (which means there is nothing to delete) else return true.

For Example, if the list **`L`** has elements **1->4->0->1->6**. Then After calling the function **`L.deleteLastOccurrence(1)`** the list will be updated as **1->4->0->6**.

I have added a generic template function.

```
template <typename T>
bool list<T>::deleteLastOccurrence(T value) {

	bool found = false;
	if (head == nullptr) //Empty List
		return found;
	else {
		SNode * current = head;  //For complete list traversal
		SNode * pcurrent = nullptr;
		SNode * toremove = nullptr; //For tracking of element
		SNode * ptoremove = nullptr;

		while (current != nullptr) // Loop to search for the last occurrence of
value
		{
			if (current-> element == value)
			{
				found = true;
				toremove = current;
				ptoremove = pcurrent;
			}
			pcurrent = current;
			current = current->next;
		}
		if (found)
		{
			if (toremove == head) // Boundary check of head node
			{
				head = head->next;
				delete toremove;
			}
			else {
				ptoremove->next = toremove->next;
				delete toremove;
			}
		}
	}

	return found;
}
```

Roll Number:_____________ Section: __________

10) Draw the contents of the hash table in the boxes below given the following conditions:

a. The size of the hash table is 7.
b. The hash function used is H(k) = k mod 7
c. Quadratic probing is used to resolve collisions.

What values will be in the hash table after the following sequence of insertions?

19, 34, 26, 16, 34, 20

Write the values in the appropriate index number of the HashMap's array given below. Also show your working (applying mod operator and quadratic probing) otherwise no marks will be given.

0	1	2	3	4	5	6
20		**26**	**16**		**19**	**34**

Working:

19 mod **7 =** **5**

34 mod **7 =** **6**

26 mod **7 = 5 Collision**

P(k, i) = i^2

New index for (26) = (5 + 1^2 **)** mod **7 = 6 filled (5 + 2**^{2} **)** mod **7 = 2 free add**

16 mod **7 = 2 Collision**

New index for (16) = (2 + 1^2 **)** mod **7 = 3 free add**

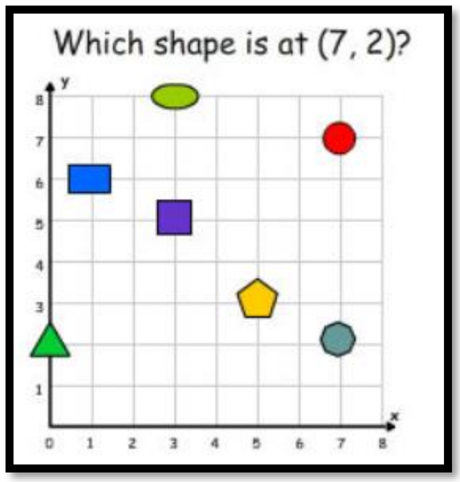
34 mod **7 = 6 Same key already inserted**

20 mod **7 = 6 Collision**

New index for (20) = (6 + 1^2 **)** mod **7 = 0 free add**

Question 2: **(Marks: 10)**

We are creating a Fun Math-App for kids to teach them about shapes and 2D maps. Currently as starter the app takes the coordinate pair (x, y) from user and tell what shape exists at that coordinate.

	**What shape is at (7, 2)** Octagon **What shape is at (0, 2)** Triangle **What shape is at (4, 2)** Nothing 😐 Input x and y coordinate from the user and output the name of the shape at that coordinate (x,y).

If we use a BST for above app, it can either search along x-axis or y-axis. So, we will use a modified BST (let call it 2D BST) for efficient searching in our Fun Math-App.

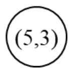
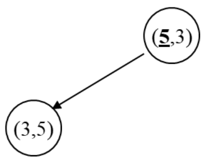
BASIC IDEA – 2D BST

To use two keys (x and y) for our 2D BST we need to use them interchangeably when descending (moving along) a 2D tree: on level 0, the *x*-coordinate is used as a discriminator (here discriminator means that we decide on the value of x-coordinate to go the left or the right subtree of the BST)

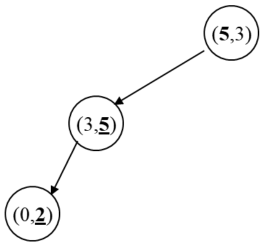
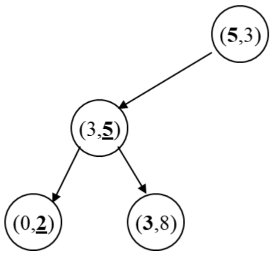
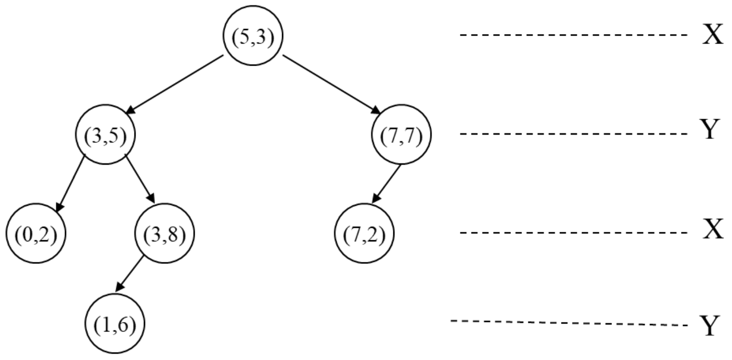
on level 1 –, the *y*-coordinate is used as a discriminator; on level 2 –, the *x*-coordinate is a discriminator, and so on.

In other words, on **even levels** the *x*-coordinate is used as a discriminator, and on **odd levels** the *y*-coordinate is used as a discriminator.

```
Class Node2dBST {
            int x, y;
            string shapeName;
            Node2dBST * lChild;
            Node2dBST * rChild;
};
```

Let's create a **2D BST** for the above map. Suppose we get the coordinate points in above map in following order (5,3), (3,5), (0,2), (3,8), (7,7), (1,6), (7,2)	
Insert first input (5,3) at level 0 (root node).	
For next input (**3**,5), start at level 0 root and use x-coordinate as discriminator to go left or right	

Roll Number:______________ Section: ___________

Next input (**0**,2), start at level 0 root and use x-coordinate as discriminator, at level 1 use y-coordinate as discriminator as 2 is less than 5 so insert to left	
Next input (**3,8**), start at level 0 and use x-coordinate as discriminator, at level 1 use y-coordinate as discriminator as 8 in (3,8) is greater than 5 in (3,5) so insert to right	
Complete 2D BST **For data in order** (5,3), (3,5), (0,2), (3,8), (7,7), (1,6), (7,2)	 (5,3) (3,5) (7,7) (0,2) (3,8) (7,2) (1,6) X Y X Y

Write a member input function in 2D BST class that takes a Node of 2D-BST as input and insert it at correct location in the tree.

```
bool BST::Insert(int x, int y, string shapeName){
      if (root == nullptr) { // Tree is empty
            Node* n = new Node(x, y, shapeName);
            root = n;
            return true;
      }
      else {
            Node* curr = root;
            Node* prev = nullptr;
            int level = 0;
            bool found = false;
            while (curr != nullptr && !found) { //iterative traversal for finding
correct insertion point
                  if (curr->x == x && curr->y == y)
                        found = true;
                  else {
                        prev = curr;
                        if (level % 2 == 0) {
                              if (curr->x > x)
                                    curr = curr->left;
                              else
                                    curr = curr->right;
                        }
                        else {
                              if (curr->y > y)
                                    curr = curr->left;
                              else
                                    curr = curr->right;

                        }
                  }
                  level++;
            }

            if (!found)//node coordiantes already in the tree no need to insert
again
            {
                  Node* n = new Node(x, y, shapeName);
                  if (level-1 % 2 == 0) {
                        if (prev->x > x)
                              prev->left = n;
                        else
                              prev->right = n;
                  }
                  else {
                        if (prev->y > y)
                              prev->left = n;
                        else
                              prev->right = n;
                  }

                  return true;
            }
      }
      return false;
}
```

Roll Number:______________ Section: ___________

Question 3: **(Marks: 5+10)**

[Smart Testing] The omicron variant of COVID-19 is spreading at an alarming pace. The minister of health of Punjab is also in a difficult situation. She has very limited budget to test COVID-19 in the people of Punjab. The officials of Punjab have collected the traces of social contacts among people in form of triples **(n_i, n_j, t_k)** which tells that person **n_i** was in contact with person **n_j** at time **t_k**. Because of the limited budget, the administration of Punjab wants to test only those individuals who are likely to be infected with COVID-19. A person is likely to be infected (susceptible) if he/she was in contact with another infected person.

For example, if person n_1 is infected at time $t_0 = 1$ and social contact trace has these triples (n_1, n_2, t_1=2) and (n_2, n_3, t_2=7) where $t_0 < t_1 < t_2$ then both n_2 and n_3 are likely to be infected. Person n_2 is likely to be infected from n_1 at t=2 and n_3 is likely to be infected from n_2 at t=7. On the other hand, if n1 is infected at t = 4 and the triples are (n_1, n_2, t_1=2) and (n_1, n_3, t_2=7), then only n_3 is likely to be infected because n_2 was in contact with n_1 before it gets infected.

As a computer science expert, your services are required. They want you to write an efficient program that can answer the following query. Given the traces of social contacts, if person **n_i** is infected at time **t_k**, output the list of people who are likely to be infected after coming in contact (directly or indirectly) with **n_i**. For simplicity, you can assume that each pair of people came in contact with each other at most once.

Model this problem as a graph problem and answer the following:

a) What are the nodes?
b) What are the edges?
c) What type of graph is this? (directed/undirected/weighted/unweighted)

a) Nodes are persons
b) Edges are formed if two persons come in direct contact with each other. Mark each edge with its contact time as its weight.
c) This will be a directed weighted graph.

Write an efficient function `FindSuseptibles` that takes the traces of social contact in the form of a graph **G**, id of infected person ***n_i***, and time of infection ***t***. This function must output the list of people who are likely to be infected after meeting ***n_i***. You can assume that graphs are implemented using adjacency list and ids of persons are from 0 to n-1, where n is the total size of the population. Also compute the time complexity of your function. You can implement function in the form of pseudo code. If you use any helper operation, then you must also give its pseudo code.

Lesser efficient solutions will be awarded less marks.

Solution:

1. Run a modified BFS on G where start node is n_i and its infection time is t.
2. Add n_i in a FIFO queue.
3. While queue is not empty extract node u from the queue.
   a. For each neighbor v of u, visit v if contact time of u and v is greater than infection time of u.
   b. Add v in the queue and set infection time of v equal to contact time of u and v.
   c. All the visited nodes are likely to be infected.

Roll Number:______________ Section: ___________

Rough Sheet

Roll Number:______________ Section: ___________

Rough Sheet

Roll Number:______________ Section: ___________

Rough Sheet

Roll Number:______________ Section: ___________

Rough Sheet