| **National University of Computer and Emerging Sciences, Lahore Campus** | | | | | |
| --- | --- | --- | --- | --- | --- |
| C:\Users\saif\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\final design.jpg | | **Course:** | **Applied Programming (AP)** | **Course Code:** | **CS-0319** |
| **Program:** | **MSCS** | **Semester:** | **Spring 2024** |
| **Deadline:** | **24-March-2023** | **Total Marks:** | **40** |
| **Section:** | **MSCS-2A** |  |  |
|  |  |  |  |
|  |  |  |  |
|  | |  | | | |
|  |  | | | | | |

**Important Instructions:**

1. Submit only one .cpp file for each question. Format: <rollno\_A1\_Qno.cpp>

2. You are not allowed to copy solutions from other students. If any sort of cheating / plagiarism is found, negative marks will be given to all students involved.

3. Late submission will result in a penalty.

**Question 1:** (heap) **[20 Marks]**

An interval heap is a binary heap in which each node contains two elements (except the last node). It is a complete

binary tree in which:

• The left element is less than or equal to the right element.

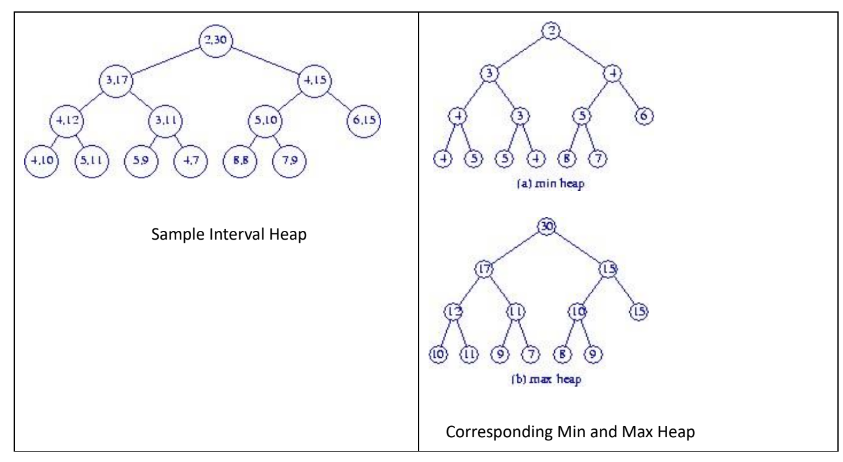
• Both the elements define a closed interval.

• Interval represented by any node except the root is a sub-interval of the parent node.

• Elements on the left hand side define a min heap.

• Elements on the right hand side define a max heap.

Below is an example of interval heap with 26 elements where interval [3, 17] is a sub-interval of its parent node i.e.

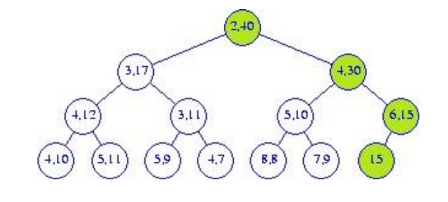
[2, 30]

Every update operation (insert or delete) must preserve all the properties mentioned above. A new node is created

in the Insertion operation if number of elements in the heap is even and no new node is created if number of

elements in the heap are odd. For example if we want to insert the element 40 in the heap given in Figure 1 then

resulting heap should be as follows:



| class IntervalHeap {  int hsize; //number of elements in heap;  int maxsize; //size of the array  int \*\*h;  public:  IntervalHeap(int s=100){  maxsize = s;  hsize = 0;  h = new int\*[2];  h[0] = new int[maxsize];//store the lower end of closed interval  h[1] = new int[maxsize];//store the higher end of closed interval  }; |
| --- |

**Implement the following functionality:**1. Write a function Insert(int d) in class IntervalHeap that inserts data element d in the interval

heap.

2. Similarly give the implementation of update, delete and display member functions for the

interval heap class.

**Question 2:** (binary tree) **[20 Marks]**

Create your own binary tree class with insertion, deletion and search functions. Implement the following methods in the class as well:

1. isSum(): bool - Given a binary tree, check if it is a sum tree or not. In a sum tree, each non-leaf node’s value is equal to the sum of all elements present in its left and right subtree. The value of a leaf node can be anything and the value of an empty child node is considered to be 0.
2. findLCA(Node\* root, Node\* x, Node\* y): Node\* Given a binary tree and two nodes, x and y, find the lowest common ancestor (LCA) of x and y in it. The solution should return null if either x or y is not the actual node in the tree. The lowest common ancestor (LCA) of two nodes x and y in a binary tree is the lowest (i.e., deepest) node that has both x and y as descendants, where each node can be a descendant of itself

