15B17CI371 – Data Structures Lab
ODD 2024
Week 12-LAB B
Practice Lab

[CO: C270.4]

C270.4	Execute the programs for different tree data structure operations like, storage, search, traverse, insertion, deletion, updating, etc. on binary trees, k-ary trees, binary search trees, AVL trees, Red-Black Trees, heap trees, B trees and B+ trees.	Apply Level (C3)

Instructions:
1. All students must save all their programs with the nomenclature (Enroll No_W12_LabB_QuestionNo.cpp). Also, store the Outputs as well.
2. Upload them as per the instructions given by your lab faculty.

Concepts: Heap – Binary heap / Binomial Heap / Fibonacci Heap

Lab Questions:

Q1. Write a program to create a max heap of integers by accepting one element at a time and by inserting it immediately into the heap. Use the array representation of heap. Display the array at the end of insertion phase.

Q2. Given a min heap having n elements stored in an array A. Code the function as per a and b and show their functionality using the main function.
- a) Code a function int findMax (int * A, int size) that should return the maximum number that can be subtracted from all the elements at deepest level without violating the min heap property.
- b) Code a function int findMinFrequency (int * A, int size) which should return the minimum no of times minimum element is extracted such that the sum of the extracted element is greater than the sum of remaining elements in the min heap.

Q3. Given a max heap having n elements stored in an array A. Code the function as per a and b and show their functionality using the main function.
- a) Code a function int findMax (int * A, int size) that should return the maximum number that can be added to all the elements at deepest level without violating the max heap property.
- b) Code a function int findMinFrequency (int * A, int size) which should return the minimum no of times maximum element is extracted such that the sum of extracted element is less than the sum of remaining elements in the min heap.

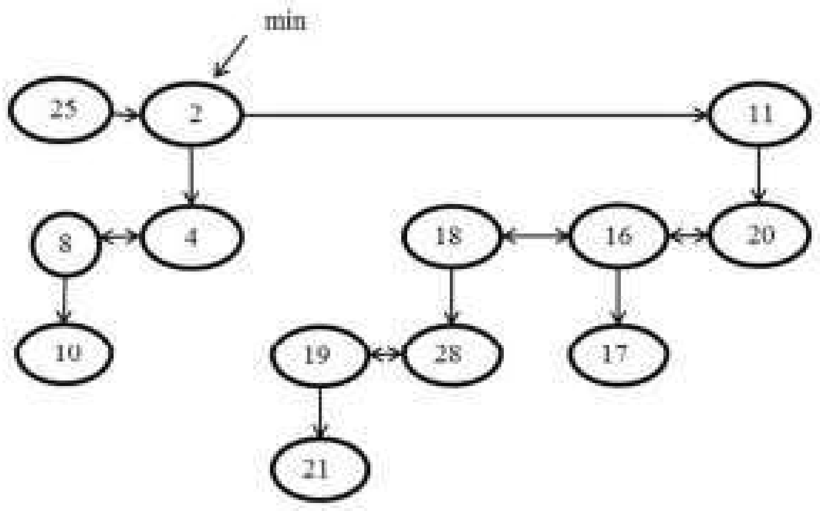
Q4. Given two binary max heaps as arrays, merge the given heaps to form a new max heap.

Q5. Given an array Arr[] of size N and an integer K, you have to choose the first two minimum elements of the array and erase them, then insert the sum of these two elements in the array until all the elements are greater than or equal to K and find the number of such operations required.

Q6. There are given N ropes of different lengths, we need to connect these ropes into one rope. The cost to connect two ropes is equal to the sum of their lengths. The task is to connect the ropes with minimum cost. Given an N size array arr[] contains the lengths of the ropes.

Q7. Implement a function that directly inserts a binomial node into a binomial heap without merging two heaps.

Q8. Write a program to print the Fibonacci heap that results from calling **FibonacciHeapExtractMin** on the Fibonacci heap shown in below figure.



Q9. The procedure to extract the Node with Minimum Key is given below.

The following procedure extracts the node with the minimum key from binomial heap H and returns a pointer to the extracted node.

```
procedure BinomialHeapExtractMin(BinomialHeap H)
        # find the root x with the minimum key in the root list of H and remove x from the root list of H
        H' ← MakeBinomialHep();
        # place the children of x in H'
        H ← BinomialHeapUnion (H, H');
        return x;
}
```

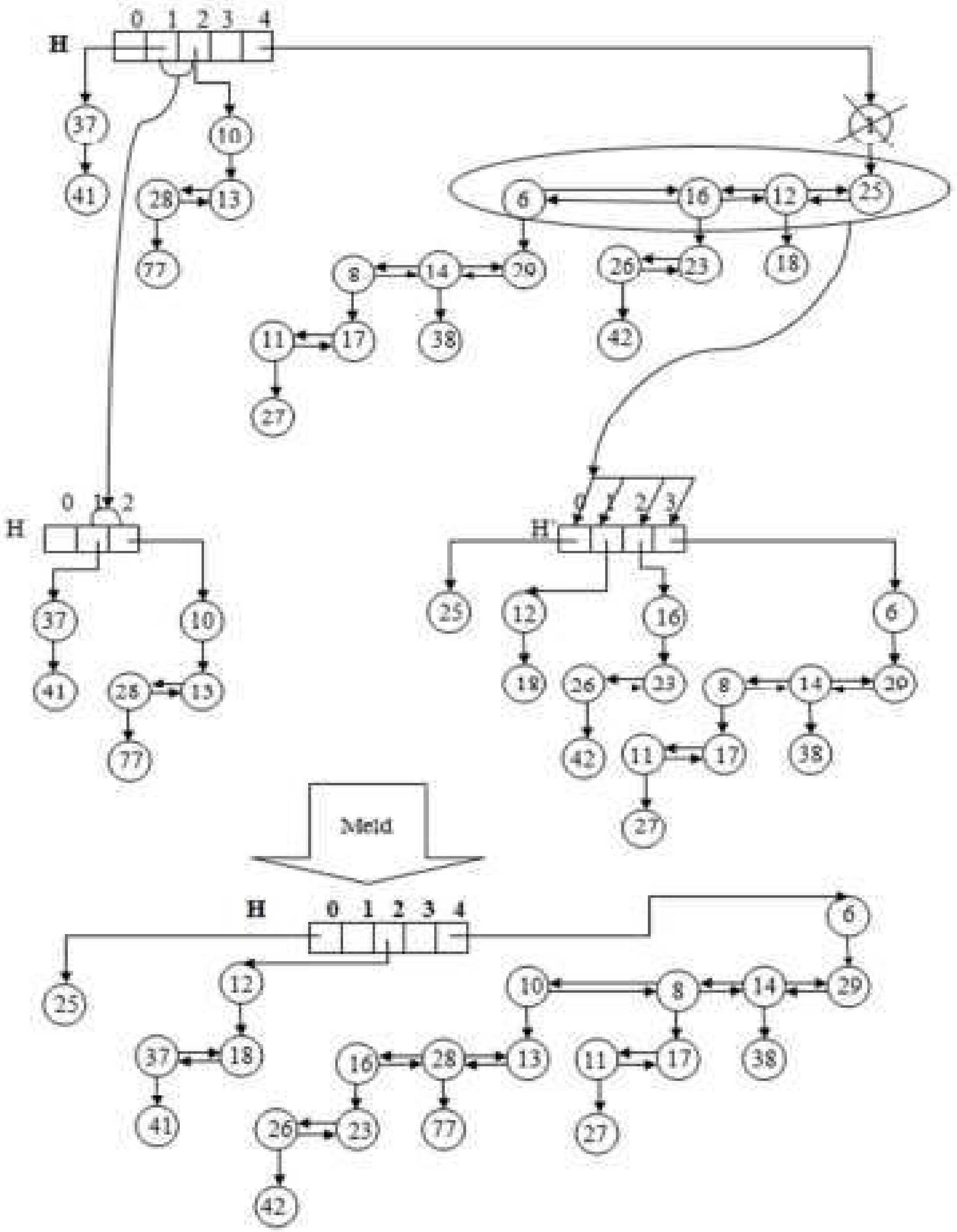
The next figure presents how works the procedure of extracting the node with the minimum key. In the initial binomial heap the minimum element is 1, the root of the **B_4** binomial tree. The goal of the procedure is to obtain a return the address of the binomial tree node containing key 1 and to obtain a correctly formed binomial heap where key 1 may not be found anymore.

Step1. The root node ***x*** containing the minimum key is obtained and returned. If the binomial heap structure keeps track of the index where the minimum element resides the ***x*** node is obtained in O(1) time. If such information is not available, all the roots of the binomial trees belonging to binomial heap H must be inspected.

Step 2. Node x is removed from the binomial heap **H**.

Step 3. All the children of the binomial tree node **x** are placed in the binomial heap **H'**.

Step 4. The current binomial heap H is merged with the binomial heap H'. Thus, it is obtained the new binomial heap H which does not contain key 1.



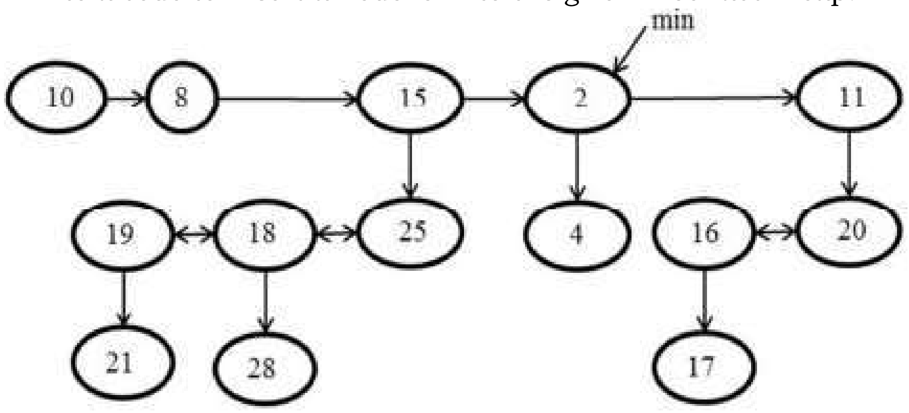
Q10. Suppose you have an array of N numbers that you want to sort smallest-to-largest. One algorithm for doing this is as follows:

a) Put all of the numbers in a min heap.
b) Repeatedly remove the min element from the heap, and store them in an array in that order.

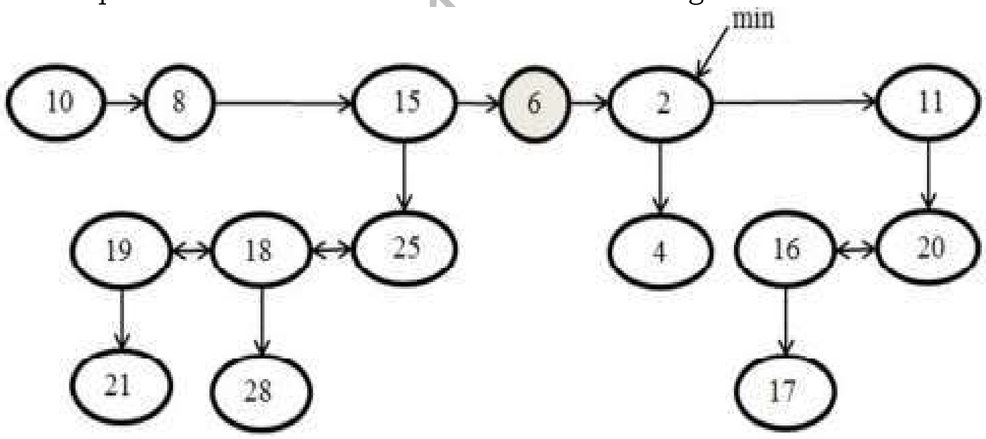
This is called *heapsort.*

Implement the heap sort algorithm with clearly showing the heapify( ) procedure in your code.

Q11. Write a code to insert a node '6' into the given Fibonacci heap.



The output after insertion will be as shown in the figure below.


min
10
8
15
6
2
11
19
18
25
4
16
20
21
28
17