ASSIGNMENT:6

AIM: Read the marks obtained by the students of second year in an online examination of a particular subject. Find out maximum and minimum marks obtained in that subject using heap data structure.

OBJECTIVE: To study and learn the concepts of heap data structure.

THEORY: Heap definition- It is a Complete (Binary) Tree with each node having HEAP PROPERTY. Elements are filled level by level from left- to-right. If A is a parent node of B, then the key (the value) of node A is ordered with respect to the key of node B with the same ordering applying across the heap.
Types of heap: 1) Min heap
2) Max heap

- MAX HEAP definition:
  - Complete (Binary) tree with the property that the **value of each node** is at least as large as the value of its children (i.e. >= value of its children)
- MIN HEAP definition:
  - Complete (Binary) tree with the property that the **value of each node** is at most as large as the value of its children (i.e. <= value of its children)

ALGORITHM: To maintain the max heap property i.e. MAXHEAPIFY

MAX-HEAPIFY(A, i, n)

1. l ← LEFT(i)
2. r ← RIGHT(i)
3. **if** l ≤ n and A[l] > A[i]
4. **then** largest ←l
5. **else** largest ←i
6. **if** r ≤ n and A[r] > A[largest]
7. **then** largest ←r
8. **if** largest ≠ i
9. **then** exchange A[i] ↔ A[largest]
10. MAX-HEAPIFY(A, largest, n)

PROGRAM:

```
#include<iostream>
#include<bits/stdc++.h>

using namespace std;
```

```
class Tree
{
    int maxHeap[50];
    int maxSize;
    int minHeap[50];
    int minSize;
    public:

        Tree()
        {
            maxSize = 0;
            minSize = 0;
        }

        void maxHeapify(int nodeIndex)
        {
            int parentIndex,temp;
            parentIndex = (nodeIndex-1)/2;
            if(maxHeap[nodeIndex]>maxHeap[parentIndex])
            {
                temp = maxHeap[nodeIndex];
                maxHeap[nodeIndex] = maxHeap[parentIndex];
                maxHeap[parentIndex] = temp;
                maxHeapify(parentIndex);
            }
        }

        void insertMaxHeap(int data)
        {
            maxSize++;
            maxHeap[maxSize-1] = data;
            maxHeapify(maxSize-1);
        }

        void minHeapify(int nodeIndex)
        {
            int parentIndex,temp;
            parentIndex = (nodeIndex-1)/2;
            if(minHeap[nodeIndex]<minHeap[parentIndex])
            {
                temp = minHeap[nodeIndex];
                minHeap[nodeIndex] = minHeap[parentIndex];
                minHeap[parentIndex] = temp;
```

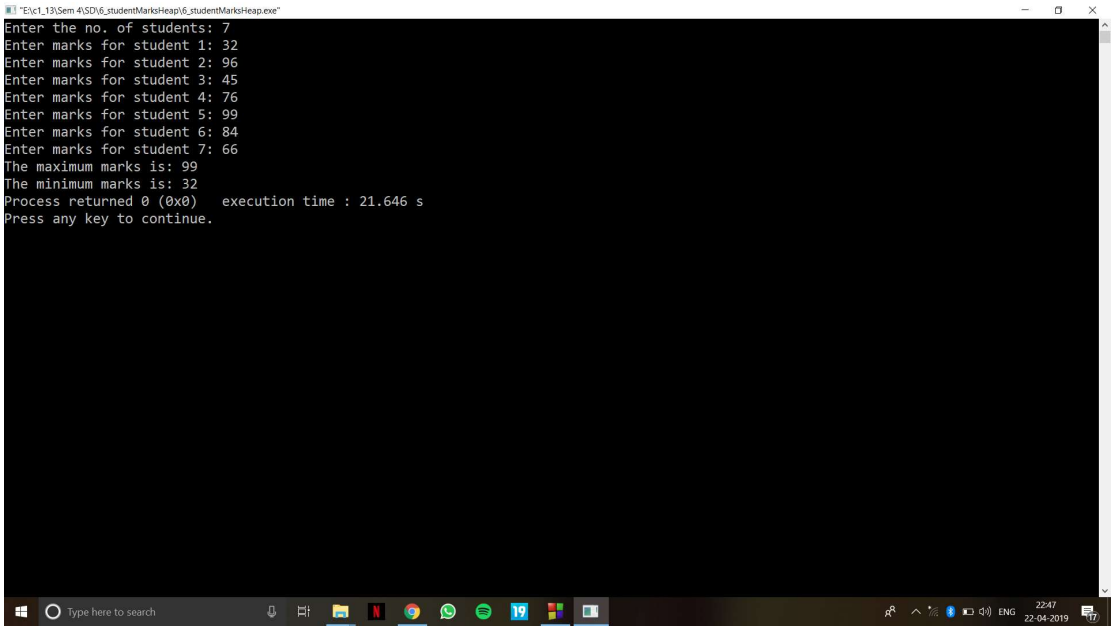
```
            minHeapify(parentIndex);
        }
    }

    void insertMinHeap(int data)
    {
        minSize++;
        minHeap[minSize-1] = data;
        minHeapify(minSize-1);
    }

    int getMaxMarks()
    {
        return maxHeap[0];
    }
    int getMinMarks()
    {
        return minHeap[0];
    }
};

int main()
{
    int student[50],n;
    Tree tree;
    cout<<"Enter the no. of students: ";
    cin>>n;
    for(int i=0;i<n;i++)
    {
        cout<<"Enter marks for student "<<i+1<<": ";
        cin>>student[i];
        tree.insertMaxHeap(student[i]);
        tree.insertMinHeap(student[i]);
    }
    cout<<"The maximum marks is: "<<tree.getMaxMarks()<<endl;
    cout<<"The minimum marks is: "<<tree.getMinMarks();
    return 0;
}
```

OUTPUT:



CONCLUSION: We successfully implemented heap data structure.