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.

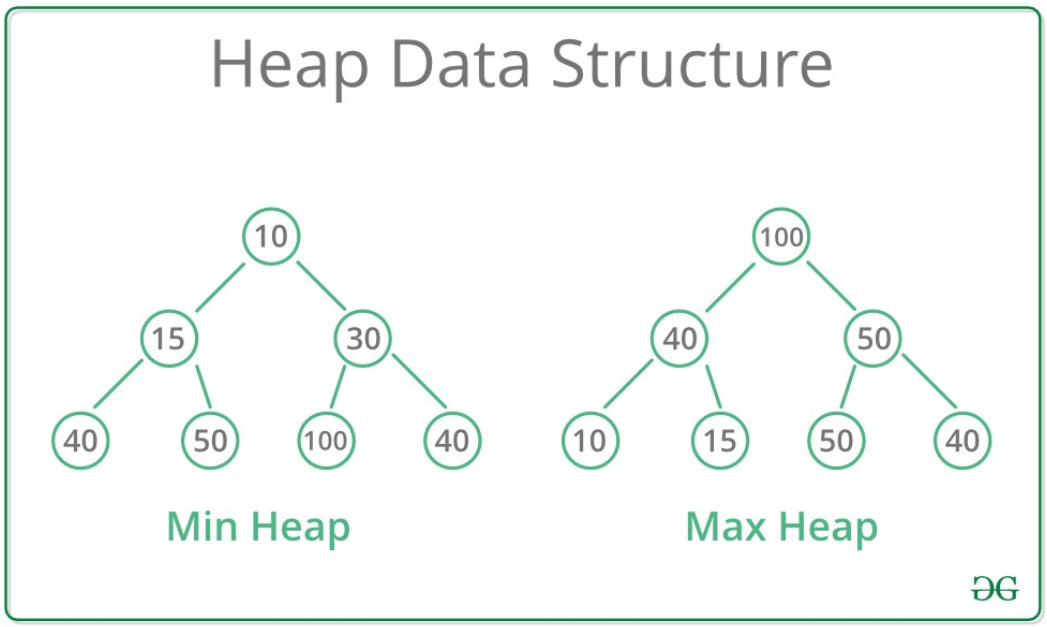
Theory:

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:

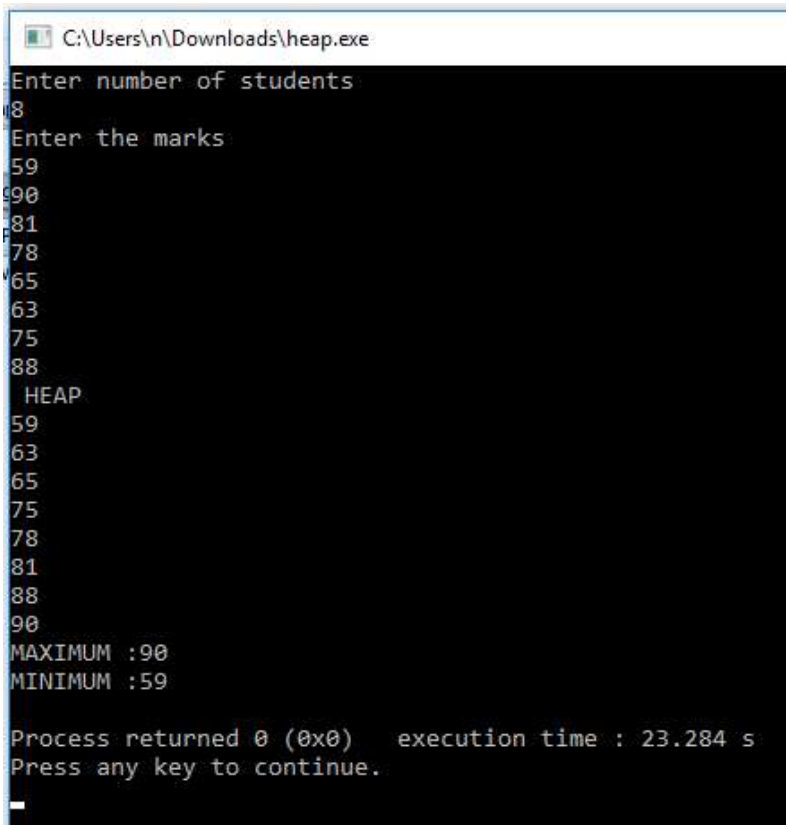
```
#include<iostream>

using namespace std;

class heap

{

public:

void printarray(int a[], int n);

void heapsort(int a[], int n);

void minimum(int a[],int n);

void maximum(int a[],int n);

};

void heapify(int a[],int n,int i);

void heap:: heapsort(int a[], int n)

 {

  for(int i=(n/2)-1; i>=0;i--)    heapify(a,n,i);
```

```
   for(int i=(n-1);i>=0;i--)
   {
     int temp= a[0];
     a[0]= a[i];
     a[i]= temp;
    heapify (a,i,0);
    }
   }
void heapify(int a[],int n, int i)
{
     int largest=i;
     int l= (2*i)+1;
     int r=(2*i)+2;
     if(l<n && a[l]>a[largest])
     largest=l;
     if(r<n && a[r]>a[largest])
     largest=r;
     if(largest!=i)
     {
     int t= a[i];
     a[i]=a[largest];
     a[largest]=t;
     heapify(a,n,largest);
     }
}
void heap:: printarray(int a[],int n)
```

```
{
    for(int i=0;i<n;i++)
      {
       cout<<a[i]<<"";
       cout<<"\n";
       }
       }
       void heap::maximum(int a[],int n)
       {
           cout<<"MAXIMUM :"<<a[n-1]<<endl;
       }
       void heap::minimum(int a[],int n)
       {
           cout<<"MINIMUM :"<<a[0]<<endl;
       }
int main()
{
  heap h;
  int a[100],n;
  cout<<"Enter number of students"<<endl;
  cin>>n;
  cout<<"Enter the marks"<<endl;
  for(int i=0;i<n;i++)
    {
    cin>>a[i];
    }
```

```
    h.heapsort(a,n);
    cout<<" HEAP"<<endl;
    h.printarray(a,n);
          h.maximum(a,n);
          h.minimum(a,n);
return 0;
 }
```

Output:



Conclusion:

We successfully implemented heap data structure. They are widely used in priority queues since they work more efficiently than linked lists