ASSIGNMENT NO.6.

Aim :-

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

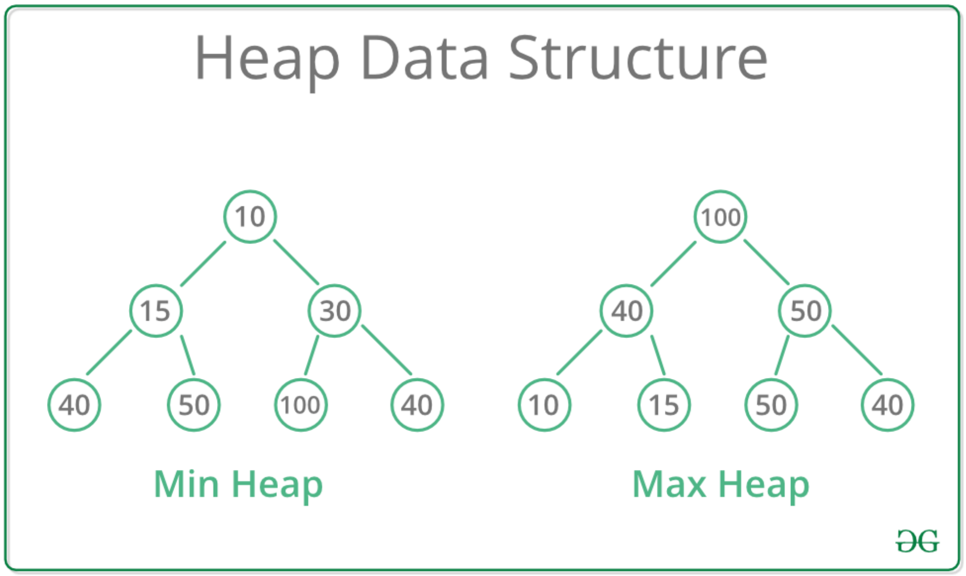
Objective:-

To study the heap data structure.

Theory:-

A Heap is a special Tree-based data structure in which the tree is a complete binary tree. Generally, Heaps can be of two types:

1. **Max-Heap**: In a Max-Heap the key present at the root node must be greatest among the keys present at all of it's children. The same property must be recursively true for all sub-trees in that Binary Tree.
2. **Min-Heap**: In a Min-Heap the key present at the root node must be minimum among the keys present at all of it's children. The same property must be recursively true for all sub-trees in that Binary Tree.



Applications:-

1) Heap Sort: Heap Sort uses Binary Heap to sort an array in O(nLogn) time.

2) Priority Queue: Priority queues can be efficiently implemented using Binary Heap because it supports insert(), delete() and extractmax(), decreaseKey() operations in O(logn) time. Binomoial Heap and Fibonacci Heap are variations of Binary Heap. These variations perform union also efficiently.
3) Graph Algorithms: The priority queues are especially used in Graph Algorithms like Dijkstra's Shortest Path and Prim's Minimum Spanning Tree.
4) Many problems can be efficiently solved using Heaps. See following for example.
a) K'th Largest Element in an array.
b) Sort an almost sorted array/
c) Merge K Sorted Arrays.

Algorithm:-

1.max heap:-

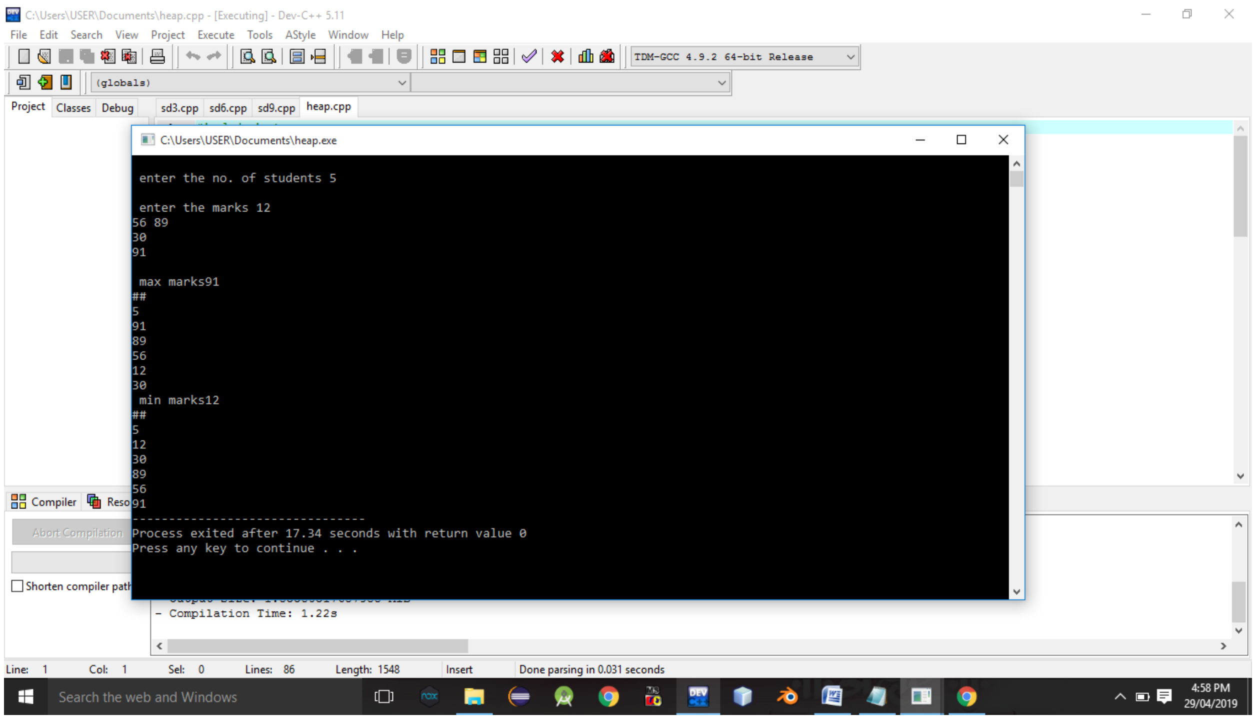
Step 1 − Create a new node at the end of heap.
Step 2 − Assign new value to the node.
Step 3 − Compare the value of this child node with its parent.
Step 4 − If value of parent is less than child, then swap them.
Step 5 − Repeat step 3 & 4 until Heap property holds.

Program Code:-

```
#include<iostream>
using namespace std;

class hp
{
   int heap[20],heap1[20],x,n1,i;
   public:
   hp()
   { heap[0]=0;  heap1[0]=0;
   }
   void getdata();
   void insert1(int heap[],int);
   void upadjust1(int heap[],int);
   void insert2(int heap1[],int);
   void upadjust2(int heap1[],int);
   void minmax();
};
void hp::getdata()
{
   cout<<"Enter the no. of students:";
   cin>>n1;
   cout<<"\n Enter their marks"<<endl;
   for(i=0;i<n1;i++)
   {   cin>>x;
       insert1(heap,x);
       insert2(heap1,x);
   }
}
void hp::insert1(int heap[20],int x)
{
   int n;
   n=heap[0];
   heap[n+1]=x;
   heap[0]=n+1;

   upadjust1(heap,n+1);
}
void hp::upadjust1(int heap[20],int i)
{
    int temp;
    while(i>1&&heap[i]>heap[i/2])
    {
       temp=heap[i];
       heap[i]=heap[i/2];
       heap[i/2]=temp;
       i=i/2;
    }
}
void hp::insert2(int heap1[20],int x)
{
   int n;
   n=heap1[0];
   heap1[n+1]=x;
   heap1[0]=n+1;

   upadjust2(heap1,n+1);
}
void hp::upadjust2(int heap1[20],int i)
{
    int temp1;
    while(i>1&&heap1[i]<heap1[i/2])
    {
       temp1=heap1[i];
       heap1[i]=heap1[i/2];
       heap1[i/2]=temp1;
       i=i/2;
    }
}
void hp::minmax()
{
   cout<<"\n Maximum marks: "<<heap[1]<<endl;

   for(i=0;i<=n1;i++)
   {   cout<<"\n"<<heap[i];  }
   cout<<"\n Minimum marks: "<<heap1[1]<<endl;
```

```
    for(i=0;i<=n1;i++)
    {   cout<<"\n"<<heap1[i];   }
}
int main()
{
  hp h;
  h.getdata();
  h.minmax();
  return 0;
}
```

Output Screenshots:-



```
 enter the no. of students 5

 enter the marks 12
56 89
30
91

 max marks91
##
5
91
89
56
12
30
 min marks12
##
5
12
30
89
56
91
--------------------------------
Process exited after 17.34 seconds with return value 0
Press any key to continue . . .
```

Conclusion:- Thus,we have studied heap data structure,