# 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.
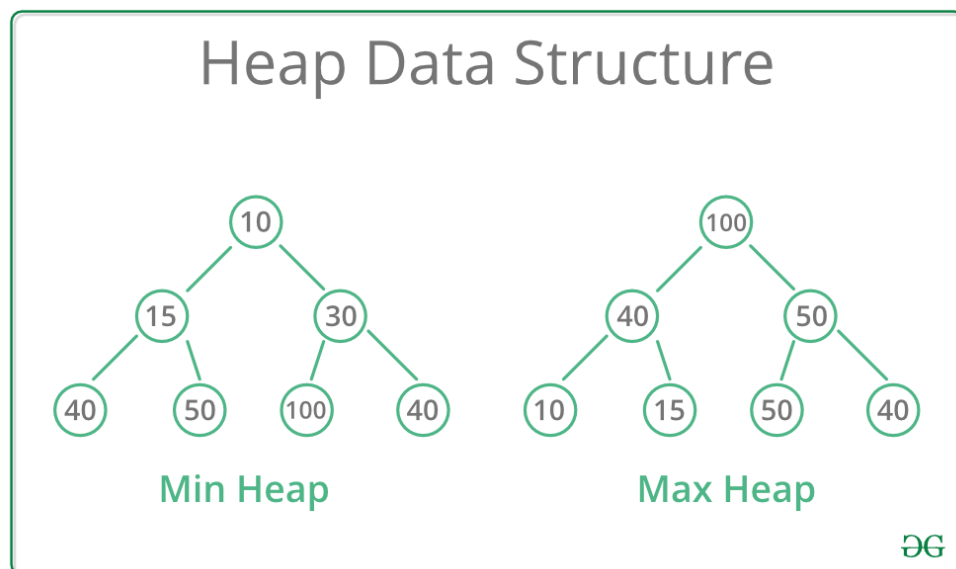
Applications:-
1.Heap sort.
2.Priority Queue.
3.Graph alogorithm shortest path e.t.c.

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:-

```cpp
#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);
```

```cpp
    void insert2(int heap1[],int);

    void upadjust2(int heap1[],int);

    void minmax();

};

void hp::getdata()

{

    cout<<"\n enter the no. of students";

    cin>>n1;

    cout<<"\n enter the marks";

    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;
```

```cpp
    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);

}
```

```cpp
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 max marks"<<heap[1];

   cout<<"\n##";

   for(i=0;i<=n1;i++)

   {   cout<<"\n"<<heap[i];  }

   cout<<"\n min marks"<<heap1[1];

   cout<<"\n##";

   for(i=0;i<=n1;i++)

   {   cout<<"\n"<<heap1[i];  }

}
```
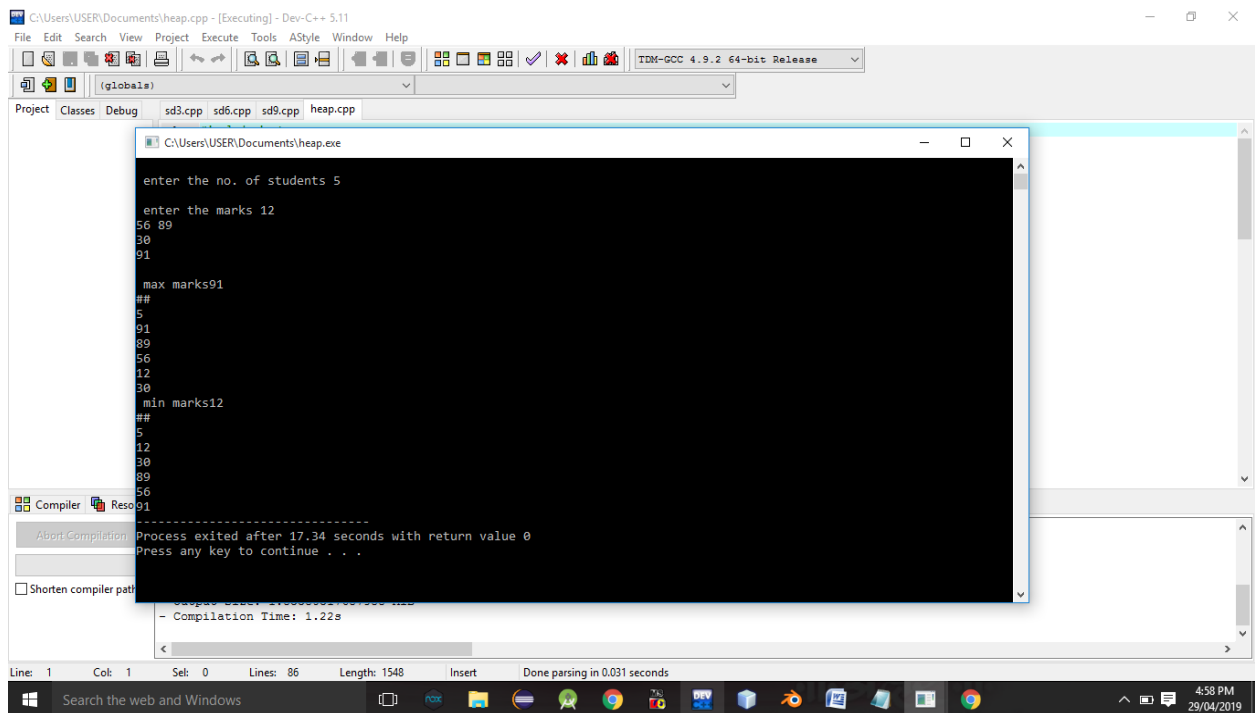
```cpp
int main()

{

  hp h;

  h.getdata();

  h.minmax();

  return 0;

}
```

Output Screenshots:-

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