NAME : Jagadeesh R

REGNO : 2021506314

OOPS Lab 9 : Implementation of Heap tree using class template.

SOURCE CODE :

```
#include<iostream>
#include<vector>
using namespace std;
template <class T>
class HeapMax{
   private:
      vector<T> heap;
   public:
     void insert(T element);
      void heapify();
      void delete_heapify();
      int size_heap();
      void display();

};
template <typename T>
void HeapMax<T>::insert(T element)
{
   heap.push_back(element);
   cout<<"Heap afer Inserting an Element :"<<element<<endl;
   display();

   heapify();
   cout<<"Heap afer Heapifing of an Elements :"<<endl;
   display();
}
template <typename T>
void HeapMax<T>::delete_heapify(){
```

```
    int size=heap.size()-1;
    int i,j;
    T temp;
    T x=heap[size];
    T val=heap[0];
    heap[0]=heap[size];
    heap[size]=val;
    i=0;
    j=i*2;
    while(j<=(size-1)){
        if(j<(size-1) && heap[j+1]>heap[j])
            j=j+1;
        if(heap[i]<heap[j]){
            temp=heap[i];
            heap[i]=heap[j];
            heap[j]=temp;
            i=j;
            j=2*j;
        }
        else
            break;
    }
    heap.pop_back();
    cout<<"Heap after Deleting of an Max element"<<endl;
    display();
    cout<<"Value Deleted "<<val<<endl;
}
template<typename T>
void HeapMax<T>::heapify(){
    int size=heap.size();
    int i=size-1;
    T temp=heap[i];
    while(i>0 && temp > heap[i/2]){
```

```
        heap[i]=heap[i/2];
        i=i/2;
    }
    heap[i]=temp;
}
template <typename T>
void HeapMax<T>::display(){
    for(int i=0;i<size_heap();i++){
        cout<<heap[i]<<" ";
    }
    cout<<endl;
}
template <typename T>
int HeapMax<T>::size_heap(){
    return (heap.size());
}
int main(){
    HeapMax<int> h;
    int temp=1;
    while(temp){
        cout<<"1.Insertion "<<endl;
        cout<<"2.Deletion "<<endl;
        cout<<"3.Size of the tree "<<endl;
        cout<<"4.Exit"<<endl;
        int choice;
        cout<<"Enter any of the choice : ";
        cin>>choice;
        cout<<endl;
        switch(choice){
            case 1:
                int key;
                cout<<"Enter the element to insert : ";
                cin>>key;
```

```
            cout<<endl;
            h.insert(key);
            break;
        case 2:
            h.delete_heapify();
            break;
        case 3:
            cout<<"Size of Heaptree is "<<h.size_heap()<<endl;
            break;
        case 4:
            temp=0;
            break;
        default:
            temp=0;
      }
   }
}
```

OUTPUT :

```
Output
/tmp/vVxZHQkadQ.o
1.Insertion
2.Deletion
3.Size of the tree
4.Exit
Enter any of the choice : 1
Enter the element to insert : 5
Heap afer Inserting an Element :5
5
Heap afer Heapifing of an Elements :
5
1.Insertion
2.Deletion
3.Size of the tree
4.Exit
Enter any of the choice : 1
Enter the element to insert : 9
Heap afer Inserting an Element :9
5 9
Heap afer Heapifing of an Elements :
9 5
1.Insertion
2.Deletion
3.Size of the tree
4.Exit
```

```
Enter any of the choice : 1
Enter the element to insert : 7
Heap afer Inserting an Element :7
9 5 7
Heap afer Heapifing of an Elements :
9 7 5
1.Insertion
2.Deletion
3.Size of the tree
4.Exit
Enter any of the choice : 1
Enter the element to insert : 3
Heap afer Inserting an Element :3
9 7 5 3
Heap afer Heapifing of an Elements :
9 7 5 3
1.Insertion
2.Deletion
3.Size of the tree
4.Exit
Enter any of the choice : 2
Heap after Deleting of an Max element
7 5 3
Value Deleted 9
```

```
1.Insertion
2.Deletion
3.Size of the tree
4.Exit
Enter any of the choice : 3
Size of Heaptree is 3
1.Insertion
2.Deletion
3.Size of the tree
4.Exit
Enter any of the choice : 4
```