.vscode\dsa\Heaps_implement.cpp

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

class heap{  // max heap
    public :
    int arr[100] ;
    int size ;

    heap() // default constructor
    {
        arr[0] = -1 ;
        size = 0 ;
    }

    void insert(int val) // T.C. = O(logn)
    {
       size = size+1 ;
       int index = size ;
       arr[index] = val ;

       while(index > 1)
       {
        int parent = index/2 ;
        if(arr[parent] < arr[index])
        {
            swap(arr[parent],arr[index]) ;
            index = parent ;
        }
        else{
            return ;
        }
       }
    }

    void print()
    {
        for(int i=1; i<=size; i++)
        {
            cout<<arr[i]<<" " ;
        }cout<<endl;
    }

    void deleteFromHeap() // T.C. = O(logn)
    // ham root node ko delete krte hai
    {
        if(size == 0)
        {
            cout<<"nothing to delete"<<endl;
            return ;
        }

        arr[1] = arr[size] ; // root node and last node ko swap krdiya ya value replace krdi
        size-- ; // remove last element

        // ab first node jo new bani hai usko shi jagah par pahunchana hai
        int i = 1 ;
        while(i < size)
        {
            int leftIndex = 2*i ;
            int rightIndex = 2*i+1 ;

            if(leftIndex < size && arr[i] < arr[leftIndex])
            {
                swap(arr[i],arr[leftIndex]) ;
                i = leftIndex ;
            }
            else if(rightIndex < size && arr[i] < arr[rightIndex])
            {
                swap(arr[i],arr[rightIndex]) ;
                i = rightIndex ;
            }
            else{
                return ;
            }
        }
    }
};

void heapify(int arr[], int n, int i) // T.C. = O(logn)
{
    int largest = i ;
    int left = 2*i ;
    int right = 2*i+1 ;

    if(left <= n && arr[largest]<arr[left])
    {
       largest = left ;
    }
    if(right <= n && arr[largest]<arr[right])
    {
        largest = right ;
    }
    if(largest != i)
    {
        swap(arr[largest],arr[i]) ;
        heapify(arr,n,largest) ;
    }
}

void heapSort(int arr[], int n) // T.C. = O(nlogn)
{
    int size = n ;
    while(size > 1)
    {
        swap(arr[size],arr[1]) ;
        size-- ;

        heapify(arr,size,1) ;
    }
}

int main()
{
    heap h ;
    h.insert(50) ;
    h.insert(55) ;
    h.insert(53) ;
    h.insert(52) ;
    h.insert(54) ;
    h.print() ;

    h.deleteFromHeap() ;
    h.print() ;

    int arr[6] = {-1,54,53,55,52,50} ;
    int n=5 ;
    for(int i=n/2;i>0;i--)
    {
        heapify(arr,n,i) ;
    }
    cout<<"printing the array now = "<<endl;
    for(int i=1;i<=n;i++)
    {
        cout<<arr[i]<<" " ;
    }cout<<endl ;

    heapSort(arr,n) ;
    cout<<"printing sorted array = "<<endl;
    for(int i=1;i<=n;i++)
    {
        cout<<arr[i]<<" " ;
    }cout<<endl ;


    cout<<"using priority queue here - "<<endl ;
    priority_queue<int> pq ; // max heap

    pq.push(4) ;
    pq.push(2) ;
    pq.push(5) ;
    pq.push(3) ;
    cout<<"element at top - "<<pq.top()<<endl ;
    pq.pop() ;
    cout<<"element at top - "<<pq.top()<<endl ;
    cout<<"size = "<<pq.size()<<endl;



    // min heap
    priority_queue<int, vector<int>, greater<int> > minHeap ;


    return 0 ;
}
```