```cpp
//*****************************************************************//
//**************string  Author = "SACHIN SAINI " ******************//
//*****************************************************************//
//Heap sort
#include<bits/stdc++.h>
using namespace std;
void build_max_heap(int [],int,int);
void heap_sort(int[],int);
void print_array(int[],int);
int main()
{
    int total_size;
    cout<<"Enter the size of array \n";
    cin>>total_size;
    cout<<"Enter the elements of array\n";
    int arr[total_size];
    for(int i=0;i<total_size;i++)
    {
        cin>>arr[i];
    }
    cout<<"Array before \n";
    print_array(arr,total_size);
    heap_sort(arr,total_size);
    cout<<"Array after \n";
    print_array(arr,total_size);
 return 0;
}
void build_max_heap(int arr[],int total_size,int parent)//building max heap
{
    int largest = parent; //take parent as largest
    int left_child = 2*parent+1;// left child will be at (2*i+1)th position
    int right_child = 2*parent+2;//right child will be at (2*i+2)th position
    //Note:: comparison is only for non leaf nodes because leaf nodes are already max heaps

    if(left_child<total_size && arr[largest]<arr[left_child])//First compare parent with its left child
    {
        largest = left_child;
    }

    if(right_child<total_size && arr[largest]<arr[right_child])//Second comparison for largest to right child
    {
        largest = right_child;
    }
    if(largest!=parent)//if current node is not max heap then swap with largest of its left/right children
    {
        swap(arr[parent],arr[largest]);
        build_max_heap(arr,total_size,largest);//call again for checking , current swapped node is following max heap condition or not
    }
}
void heap_sort(int arr[],int total_size)
{
    for(int i=(total_size/2)-1;i>=0;i--)//leaf nodes are already sorted then comparison will be cone only for non leaf nodes
        //and for complete binary tree or heap tree there are total half of elements are non leaf nodes so comparison is going from size/2 to 1st element back
    {
        build_max_heap(arr,total_size,i);
    }
    //now extract element one by one from heap
    for(int i=total_size-1;i>=0;i--)
    {
        //move root to the end
        swap(arr[0],arr[i]);

        //call build_max_function on reduced heap
        build_max_heap(arr,i,0);
    }
}
void print_array(int arr[],int total_size)
{
    for(int i=0;i<total_size;i++)
    {
        cout<<arr[i]<<" ";
    }
    cout<<endl;
}

```