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

// To heapify a subtree rooted with node i
// 12,11,13,5,6,7 and n=6 and i=2 i=1 i=0 // 7,11,12,5,6,13 and n=5 and i=0
void heapify(int arr[], int n, int i)
{
 int largest = i; // Initialize largest as root // largest=2 // largest=1 // largest=0 // largest=0
 int l = 2*i + 1; // left child // l = 2*2+1 = 5 // l=3 // l=1 // l=1
 int r = 2*i + 2; // right child // r = 2*2+2 = 6 // r=4 // r=2 // r=2

 // If left child is larger than root
 if(l < n && arr[l] > arr[largest]) // 5<6 and arr[5]=7 > arr[2]=13 // 3<6 and 5>11 // 1<6 and 11>12 // 1<6 and 11>13
 // 1<5 & 11>7
  largest = l; // largest=1

 // If right child is larger than largest so far
 if(r < n && arr[r] > arr[largest]) // 6<6 // 4<6 and 6>11 // 2<6 and 13>12 // 2<6 and 12>13 // 2<5 & 12>11
  largest = r; // largest=2 // largest=2

 // If largest is not root
 if(largest != i) // 2==2 // 1==1 // 2!=0 // 0==0 // 2!=0
 {
  swap(arr[i], arr[largest]); // 13,11,12,5,6,7 // 12,11,7,5,6,12

  // Recursively heapify the sub-tree
  heapify(arr, n, largest);
 }
}

// 12,11,13,5,6,7 and 6
void heapSort(int arr[], int n)
{
 // Build heap
 for(int i = n / 2 - 1; i >= 0; i--) // 2 to 0
  heapify(arr, n, i);

 // One by one extract an element from heap
 for(int i = n-1; i > 0; i--) // 5 to 1
 {
  // Move current root to end // i=5
  swap(arr[0], arr[i]); // 7,11,12,5,6,13

  // call max heapify on the reduced heap // i=5
  heapify(arr, i, 0); //
 }
}

int main()
{
 int arr[] = {12,11,13,5,6,7};
 int n = sizeof(arr)/sizeof(arr[0]);

 heapSort(arr, n);
```

```
 cout<<"Sorted array is \n";
 for(int i = 0; i < n; ++i)
  cout<<arr[i]<<" ";
 cout<<"\n";
}
```