#include <iostream>  
using namespace std;  
  
class Heap{  
private:  
 int length;  
 int arr[100];  
 int heapSize;  
  
public:  
 void print() {  
 for(int i=0; i<length; i++)  
 cout<<arr[i]<<" ";  
 cout<<endl;  
 }  
  
 Heap(int \*arr, int length) {  
 for(int i=0; i<length; i++)  
 this->arr[i] = arr[i];  
 this->length=length;  
 this->heapSize=length;  
 }  
  
 int Parent(int i) {  
 return (i-1)/2;  
 }  
 int Left(int i) {  
 return (i\*2)+1;  
 }  
 int Right(int i) {  
 return (i\*2)+2;  
 }  
  
 void MaxHeapify(int i) {  
 int largest = i;  
 int left = Left(i);  
 int right = Right(i);  
  
 if(left < heapSize && arr[left] > arr[largest])  
 largest=left;  
 if(right < heapSize && arr[right] > arr[largest])  
 largest=right;  
 if(largest!=i) {  
 swap(arr[i], arr[largest]);  
 MaxHeapify(largest);  
 }  
 }  
  
 void BuildMaxHeap() {  
 for(int i=Parent(length-1); i>=0; i--)  
 MaxHeapify(i);  
 }  
  
 void HeapSort() {  
 BuildMaxHeap();  
 for(int i=0; i<length; i++) {  
 swap(arr[0], arr[heapSize - 1]);  
 heapSize--;  
 MaxHeapify(0);  
 }  
 }  
};  
  
int main() {  
 int arr[20] = {12,32,45765,2,55,8,2,55,58,154,7,9,26,4,9,2,675,4265,545,2};  
 Heap obj(arr, 20);  
 obj.HeapSort();  
 obj.print();  
  
 return 0;  
}

BuildMaxHeap – to build heap -> O(n);

MaxHeapify – to heapify -> O(log n); Because HeapSort is a binary algorithm that builds heap.

Overall Time Complexity: O(n log n);

Heapsort is an efficient, unstable sorting algorithm with an average, best-case, and worst-case time complexity of O(n log n). Heapsort is significantly slower than Quicksort and Merge Sort