```
void heapify(int arr[], int n, int root)
{
        int largest = root;   // root is the largest element
        int l = 2 * root + 1; // left = 2*root + 1
        int r = 2 * root + 2; // right = 2*root + 2
        // If left child is larger than root
        if ((l < n) && (arr[l] > arr[largest])) {
                largest = l;
        }
        // If right child is larger than largest so far
        if ((r < n) && (arr[r] > arr[largest])) {
                largest = r;
        }
        // If largest is not root
        if (largest != root) {
                //swap root and largest
                swap(arr[root], arr[largest]);
                // Recursively heapify the sub-tree
                heapify(arr, n, largest);
        }
}
```

```
void heapSort(int arr[], int n)
{
        for (int i = n / 2 - 1; i >= 0; i--) {   // build heap
                heapify(arr, n, i);
        }
        for (int i = n - 1; i >= 0; i--) {        // extracting elements
                                                 //  from heap one by one
                swap(arr[0], arr[i]);            // Move current root to end
                heapify(arr, i, 0);              // again call max heapify on
                                                 // the reduced heap
        }
}
```