Question 1.)

package lcolemancsc310assign5;

// Java program to implement a Heap with remove\_max() capabilities

public class LColemanCSC310Assign5 {

public void sort(int arr[]){

int n = arr.length;

// Build heap (rearrange array)

for (int i = n / 2 - 1; i >= 0; i--)

heapify(arr, n, i);

// Extract one element at a time from heap

for (int i=n-1; i>=0; i--){

// Move root to end

int temp = arr[0];

arr[0] = arr[i];

arr[i] = temp;

heapify(arr, i, 0);

}

}

void heapify(int arr[], int n, int i){

int largest = i;

int l = 2\*i + 1;

int r = 2\*i + 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 != i){

int swap = arr[i];

arr[i] = arr[largest];

arr[largest] = swap;

// Recursively heapify the affected sub-tree

heapify(arr, n, largest);

}

}

// A function to print array

static void printArray(int arr[]){

int n = arr.length;

for (int i=0; i<n; ++i)

System.out.print(arr[i]+" ");

System.out.println();

}

public static void main(String args[]) {

int arr[] = {9,7,5,2,6,4};

int n = arr.length;

LColemanCSC310Assign5 ob = new LColemanCSC310Assign5();

ob.sort(arr);

System.out.println("Sorted array is");

printArray(arr);

}

}

Question 2.)

package lcolemancsc310assign5pt2;

class binHeap{

public int capacity;

public int [] mH;

public int currentSize;

public binHeap(int capacity){

this.capacity=capacity;

mH = new int [capacity+1];

currentSize =0;

}

public void createHeap(int [] arrA){

if(arrA.length>0){

for(int i=0;i<arrA.length;i++){

insert(arrA[i]);

}

}

}

public void print(){

for(int i=1;i<mH.length;i++){

System.out.print(" " + mH[i]);

}

System.out.println("");

}

public void insert(int x) {

if(currentSize==capacity){

System.out.println("heap is full");

return;

}

currentSize++;

int idx = currentSize;

mH[idx] = x;

upHeap(idx);

}

public void upHeap(int pos) {

int parentIdx = pos/2;

int currentIdx = pos;

while (currentIdx > 0 && mH[parentIdx] > mH[currentIdx]) {

swap(currentIdx,parentIdx);

currentIdx = parentIdx;

parentIdx = parentIdx/2;

}

}

public int find\_Min() {

int min = mH[1];

mH[1] = mH[currentSize];

mH[currentSize] = 0;

downheap(1);

currentSize--;

return min;

}

public void downheap(int k) {

int smallest = k;

int leftChildIdx = 2 \* k;

int rightChildIdx = 2 \* k+1;

if (leftChildIdx < size() && mH[smallest] > mH[leftChildIdx]) {

smallest = leftChildIdx;

}

if (rightChildIdx < size() && mH[smallest] > mH[rightChildIdx]) {

smallest = rightChildIdx;

}

if (smallest != k) {

swap(k, smallest);

downheap(smallest);

}

}

public void swap(int a, int b) {

int temp = mH[a];

mH[a] = mH[b];

mH[b] = temp;

}

public boolean is\_Empty() {

return currentSize == 0;

}

public int size(){

return currentSize;

}

}

public class LColemanCSC310Assign5pt2 {

public static void main(String args[]){

int arrA [] = {5,7,3,11};

System.out.print("Original Array: ");

for(int i=0;i<arrA.length;i++){

System.out.print(" " + arrA[i]);

}

System.out.println();

binHeap m = new binHeap(arrA.length);

System.out.print("Min-Heap : ");

m.createHeap(arrA);

m.print();

System.out.print("Find Min :");

for(int i=0;i<arrA.length;i++){

System.out.print(" " + m.find\_Min());

}

System.out.println();

}

}