**LAB 11:**

**Question 01:**

import java.util.Scanner;

public class Question\_number1 {

int[] arr ;

int size;

int maxsize;

Question\_number1(int data){

arr = new int[data+1];

maxsize = data;

}

public static void main(String[] args) {

Question\_number1 q1 = new Question\_number1(8);

q1.insert(14);

q1.insert(5);

q1.insert(8);

q1.insert(17);

q1.insert(3);

q1.insert(21);

q1.printing();

System.out.println("Enter new value: ");

Scanner sc = new Scanner(System.in);

q1.insert(sc.nextInt());

q1.printing();

System.out.println("\nFor checking leaf node Enter node's position : ");

q1.childs(sc.nextInt());

System.out.println("====================================================");

q1.MinheaptoMaxheap(7);

System.out.println("....With respect to Max-Heap....");

q1.printing();

}

public void childs (int pos){

if (pos<maxsize/2){

System.out.print("\nLeft child: ");

System.out.print(arr[pos\*2]);

System.out.print(" || Right child: ");

System.out.println(arr[pos\*2 +1]);

}else {

System.out.println("Node is leaf node");

}

}

public int parent(int position){

return position/2;

}

// printing

public void printing(){

for (int i = 1; i < maxsize; i++) {

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

}

}

// inserting

public void insert(int data){

if (size>maxsize){

return;

}

arr[++size] = data;

int current = size;

// if current value is less

while (arr[current]<arr[parent(current)]){

swap(current,parent(current));

current = parent(current);

}

}

public void swap(int pos1,int pos2){

int temp = arr[pos1];

arr[pos1]= arr[pos2];

arr[pos2] = temp;

}

public void MinheaptoMaxheap(int i) {

if (i < 1) {

return;

}else {

int curr = i;

while (arr[curr] > arr[parent(curr)]) {

swap(curr, parent(curr));

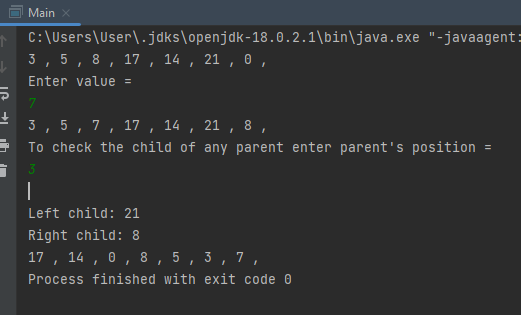
curr = parent(curr);

}

MinheaptoMaxheap(--i);

}

} }



**Question 02:**

public class Lab11Task2 {

int[] PriQue;

int last=-1;

Lab11Task2(int size){

PriQue=new int[size];

}

public void heapify(int[] heap,int capacity,int i){

int largest=i;

int left=2\*i+1;

int right=2\*i+2;

if (left<=capacity&&heap[left]>heap[largest]){

largest=left;

}

if (right<=capacity&&heap[right]>heap[largest]){

largest=right;

}

if (largest!=i){

int swap=heap[i];

heap[i]=heap[largest];

heap[largest]=swap;

heapify(heap,capacity,largest);

}

}

public void insert(int data){

if (last==PriQue.length-1){

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

return;

}

if (last<0){

PriQue[++last]=data;

return;

}

PriQue[++last]=data;

for (int i=last/2;i>=0;i--){

heapify(PriQue,last,i);

}

}

public int delete(){

int temp=PriQue[0];

PriQue[0]=PriQue[last];

PriQue[last--]=0;

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

heapify(PriQue,last,i);

}

return temp;

}

public static void transfer(Lab11Task2 pq1,Lab11Task2 pq2){

for (int i=0;i<pq1.PriQue.length;i++){

pq2.insert(pq1.delete());

}

}

public void print(int[] arr){

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

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

}

}

public static void main(String[] args) {

Lab11Task2 pq1=new Lab11Task2(5);

Lab11Task2 pq2=new Lab11Task2(5);

pq1.insert(5);

pq1.insert(8);

pq1.insert(6);

pq1.insert(4);

pq1.insert(3);

System.out.println("Elements of the first queue");

pq1.print(pq1.PriQue);

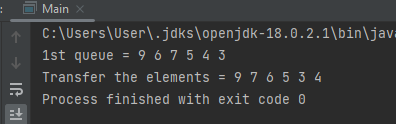
System.out.println();

System.out.println("Transfer the elements ");

transfer(pq1,pq2);

pq2.print(pq2.PriQue);

} }



**Question 03:**

import java.util.\*;

public class Qno3\_Lab11 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int arr[] = new int[3];

int str[] = new int[6];

int j=0;

System.out.println("Enter");

Job jb1 = new Job(arr.length);

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

str[i]=sc.nextInt();

if(str[i]<60){

arr[j]=str[i];

j++;

}

else{

jb1.Max\_Priority(str[i]);

}

}

System.out.println("Maximum Prioriy");

jb1.Max\_Print();

Job jb = new Job(arr.length,arr);

jb.MIn\_Priority();

System.out.println("Maximum Prioriy");

jb.Min\_Print();

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

if(jb1.Max\_Size()>0){

System.out.println("Delete Max: "+jb1.Max\_Delete());

}

else{

System.out.println("Delete Min: "+jb.Min\_Delete());

}

}

}

}

class Job{

int arr1[];

int arr2[];

int size1;

int n1=0;

int size2;

int n2=0;

public Job(int size){

this.size1=size;

arr1=new int[size1];

}

public Job(int size,int arr[]){

this.size2=arr.length;

this.arr2=arr;

}

public void Max\_Heapify(int i){

int largest=i;

int left=i\*2;

int right=i\*2+1;

if(left<size1 && arr1[left]>arr1[largest]){

largest=left;

}

if(right<size1 && arr1[right]>arr1[largest]){

largest=right;

}

if(largest!=i){

int temp =arr1[i];

arr1[i]=arr1[largest];

arr1[largest]=temp;

Max\_Heapify(largest);

}

}

public void Max\_Priority(int data){

if(n1==size1-1){

System.out.println();

}

arr1[n1]=data;

n1++;

int p=(n1/2);

for(int i=p;i>=0;i--){

Max\_Heapify(i);

}

}

public int Max\_Size(){

return n1;

}

public int Max\_Delete(){

int max=arr1[0];

arr1[0]=arr1[n1-1];

n1=n1-1;

Max\_Heapify(0);

return max;

}

public void Max\_Print(){

for(int i=0;i<n1;i++){

System.out.print(arr1[i]+" ");

}

System.out.println();

}

public void Min\_Heapify(int i){

int smallest=i;

int left=i\*2+1;

int right=i\*2+2;

if(left<size2 && arr2[left]<arr2[smallest]){

smallest=left;

}

if(right<size2 && arr2[right]<arr2[smallest]){

smallest=right;

}

if(smallest!=i){

int temp =arr2[i];

arr2[i]=arr2[smallest];

arr2[smallest]=temp;

Min\_Heapify(smallest);

}

}

public void MIn\_Priority(){

int p=(size2/2)-1;

for(int i=p;i>=0;i--){

Min\_Heapify(i);

}

}

public int Min\_Delete(){

int min=arr2[0];

arr2[0]=arr2[size2-1];

Min\_Heapify(0);

return min;

}

public int Min\_Size(){

return n2;

}

public void Min\_Print(){

for(int i=0;i<size2;i++){

System.out.print(arr2[i]+" ");

}

System.out.println();

} }

