Assessment

# Q1.

## Bubble sort:

import java.io.\*;

class sorting1 {

static void bubbleSort(int arr[], int n){

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

for(int j = 0; j < n - i - 1; j++){

if( arr[j] > arr[j + 1]){

int temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

}

public static void main (String[] args) {

int a[] = {2, 1, 4, 3};

bubbleSort(a, 4);

for(int i = 0; i < 4; i++){

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

}

}

## Selection Sort:

import java.io.\*;

class SelectionSort

{

static void selectionSort(int arr[], int n){

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

int min\_ind = i;

for(int j = i + 1; j < n; j++){

if(arr[j] < arr[min\_ind]){

min\_ind = j;

}

}

int temp = arr[i];

arr[i] = arr[min\_ind];

arr[min\_ind] = temp;

}

}

public static void main (String[] args) {

int a[] = {2, 1, 4, 3};

selectionSort(a, 4);

for(int i = 0; i < 4; i++){

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

}

}

}

## Insersion Sort:

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class InsertionSort

{

public static void main (String[] args)

{

int arr[] = new int[]{50,20,40,60,10,30};

int n = arr.length;

iSort(arr,n);

for(int x:arr)

System.out.print(x+" ");

}

static void iSort(int arr[],int n){

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

int key = arr[i];

int j=i-1;

while(j>=0 && arr[j]>key){

arr[j+1]=arr[j];

j--;

}

arr[j+1]=key;

}

}

}

## Merge Sort:

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class MergeSort

{

public static void main (String[] args)

{

int a[] = new int[]{10,5,30,15,7};

int l=0,r=4;

mergeSort(a,l,r);

for(int x: a)

System.out.print(x+" ");

}

static void merge(int arr[], int l, int m, int h){

int n1=m-l+1, n2=h-m;

int[] left=new int[n1];int[]right=new int[n2];

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

left[i]=arr[i+l];

for(int j=0;j<n2;j++)

right[j]=arr[m+1+j];

int i=0,j=0,k=l;

while(i<n1 && j<n2){

if(left[i]<=right[j])

arr[k++]=left[i++];

else

arr[k++]=right[j++];

}

while(i<n1)

arr[k++]=left[i++];

while(j<n2)

arr[k++]=right[j++];

}

static void mergeSort(int arr[],int l,int r){

if(r>l){

int m=l+(r-l)/2;

mergeSort(arr,l,m);

mergeSort(arr,m+1,r);

merge(arr,l,m,r);

}

}

}

## Heap Sort:

import java.util.\*;

import java.io.\*;

import java.lang.\*;

public class HeapSort

{

public void buildheap(int arr[],int n){

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

heapify(arr, n, i);

}

public void sort(int arr[])

{

int n = arr.length;

buildheap(arr,n);

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

{

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 (l < n && arr[l] > arr[largest])

largest = l;

if (r < n && arr[r] > arr[largest])

largest = r;

if (largest != i)

{

int swap = arr[i];

arr[i] = arr[largest];

arr[largest] = swap;

heapify(arr, n, largest);

}

}

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[] = {12, 11, 13, 5, 6, 7};

int n = arr.length;

HeapSort ob = new HeapSort();

ob.sort(arr);

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

printArray(arr);

}

}

## Quick Sort:

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class Quick

{

public static void main (String[] args)

{

int arr[] = new int[]{8,4,7,9,3,10,5};

int n = arr.length;

qSort(arr,0,n-1);

for(int x: arr)

System.out.print(x+" ");

}

static int partition(int arr[], int l, int h)

{

int pivot=arr[l];

int i=l-1,j=h+1;

while(true){

do{

i++;

}while(arr[i]<pivot);

do{

j--;

}while(arr[j]>pivot);

if(i>=j)return j;

int temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

static void qSort(int arr[],int l,int h){

if(l<h){

int p=partition(arr,l,h);

qSort(arr,l,p);

qSort(arr,p+1,h);

}

}

}

## Quene:

import java.util.\*;

class quene {

public static void main (String[] args) {

// Queue<Integer> q = new LinkedList<Integer>();

Queue<Integer> q = new ArrayDeque<Integer>();

q.offer(10);

q.offer(20);

q.offer(30);

System.out.println(q.peek());

System.out.println(q.poll());

System.out.println(q.peek());

}

}

## Linkedlist single:

import java.util.\*;

import java.io.\*;

import java.lang.\*;

class Node{

int data;

Node next;

Node(int x){

data=x;

next=null;

}

}

class Test {

static Node insertEnd(Node head, int x){

Node temp=new Node(x);

if(head==null)return temp;

Node curr=head;

while(curr.next!=null){

curr=curr.next;

}

curr.next=temp;

return head;

}

public static void main(String args[])

{

Node head=null;

head=insertEnd(head,10);

head=insertEnd(head,20);

head=insertEnd(head,30);

printlist(head);

}

public static void printlist(Node head){

Node curr=head;

while(curr!=null){

System.out.print(curr.data+" ");

curr=curr.next;

}

}

}

## Double linkedlist:

import java.util.\*;

import java.io.\*;

import java.lang.\*;

class Node{

int data;

Node prev;

Node next;

Node(int d){

data=d;

prev=null;

next=null;

}

}

class Test {

public static void main(String args[])

{

Node head=new Node(10);

Node temp1=new Node(20);

Node temp2=new Node(30);

head.next=temp1;

temp1.prev=head;

temp1.next=temp2;

temp2.prev=temp1;

printlist(head);

}

public static void printlist(Node head){

Node curr=head;

while(curr!=null){

System.out.print(curr.data+" ");

curr=curr.next;

}System.out.println();

}

}