**Company Preparation Session -**

**1) Which of the following is true for computation time in insertion, deletion and finding maximum and minimum element in a sorted array ?**

A.Insertion – 0(1), Deletion – 0(1), Maximum – 0(1), Minimum – 0(l)

B.Insertion – 0(1), Deletion – 0(1), Maximum – 0(n), Minimum – 0(n)

C.Insertion – 0(n), Deletion – 0(n), Maximum – 0(1), Minimum – 0(1)

D.Insertion – 0(n), Deletion – 0(n), Maximum – 0(n), Minimum – 0(n)

**Ans: C**

**2) What is the best time complexity of bubble sort?**

A.N^2

B.NlogN

C.N

D.N(logN)^2

**Ans: N**

**3) Which of the following is not a stable sorting algorithm in its typical implementation.**

(A) Insertion Sort

(B) Merge Sort

(C) Quick Sort

(D) Bubble Sort

**Ans: C**

4) **Running merge sort on an array of size n which is already sorted is**

a) O(n)

b) O(nlogn)

c) O(n2)

d)None

**Ans: b**

5) **If the given input array is sorted or nearly sorted, which of the following algorithm gives the best performance?**

a) Insertion sort

b) Selection sort

c) Quick sort

d) Merge sort

**Ans: a**

6) **Which of the following sorting algorithm has the running time that is least dependant on the initial ordering of the input?**

a) Insertion sort

b) Quick sort

c) Merge sort

d) Selection sort

**Ans: a**

**7) Which of the following algorithm design technique is used in the quick sort algorithm?**

a) Dynamic programming

b) Backtracking

c) Divide-and-conquer

d) Greedy method

**Ans: C**

**8) For merging two sorted lists of size m and n into sorted list of size m+n, we require comparisons of**

a) O(m)

b) O(n)

c) O(m+n)

d) O(logm + logn)

**Ans: C**

**9) A sorting technique is called stable if it**

a) Takes O(nlogn) times

b) Maintains the relative order of occurrence of non-distinct elements

c) Uses divide-and-conquer paradigm

d) Takes O(n) space

**Ans: B**

**10) What is the advantage of bubble sort over other sorting techniques?**

a) It is faster

b) Consumes less memory

c) Detects whether the input is already sorted

d) All of the mentioned

**Ans: C**

**11) How can you improve the best case efficiency in bubble sort? (The input is already sorted)**

a)

**boolean** swapped = **false**;

**for**(**int** j=arr.length-1; j>=0 && swapped; j--)

{

swapped = **true**;

**for**(**int** k=0; k<j; k++)

{

**if**(arr[k] > arr[k+1])

{

**int** temp = arr[k];

arr[k] = arr[k+1];

arr[k+1] = temp;

swapped = **false**;

}

}

}

b)

**boolean** swapped = **true**;

**for**(**int** j=arr.length-1; j>=0 && swapped; j--)

{

swapped = **false**;

**for**(**int** k=0; k<j; k++)

{

**if**(arr[k] > arr[k+1])

{

**int** temp = arr[k];

arr[k] = arr[k+1];

arr[k+1] = temp;

}

}

}

c)

**boolean** swapped = **true**;

**for**(**int** j=arr.length-1; j>=0 && swapped; j--)

{

swapped = **false**;

**for**(**int** k=0; k<j; k++)

{

**if**(arr[k] > arr[k+1])

{

**int** temp = arr[k];

arr[k] = arr[k+1];

arr[k+1] = temp;

swapped = **true**;

}

}

}

d)

**boolean** swapped = **true**;

**for**(**int** j=arr.length-1; j>=0 && swapped; j--)

{

**for**(**int** k=0; k<j; k++)

{

**if**(arr[k] > arr[k+1])

{

**int** temp = arr[k];

arr[k] = arr[k+1];

arr[k+1] = temp;

swapped = **true**;

}

}

}

**Ans: C**

**12) The given array is arr = {1,2,4,3}. Bubble sort is used to sort the array elements. How many iterations will be done to sort the array with improvised version?**

a) 4

b) 2

c) 1

d) 0

**Ans: B**

**13) What is the best case complexity of QuickSort?**

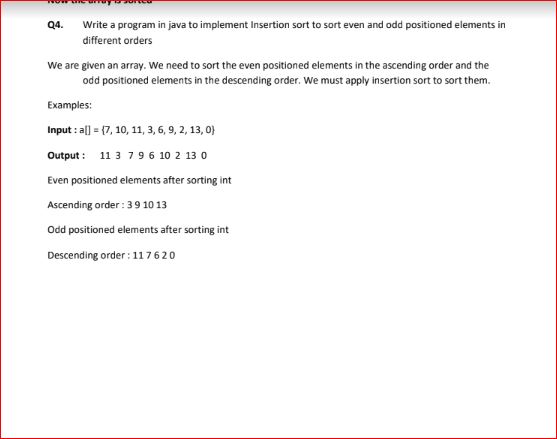
A. O(nlogn)

B. O(logn)

C. O(n)

D. O(n2)

**Ans: A**

****

**Soln:-**

public class InsertionsortEvenOdd {

static void EvenOddInsertionSort(int arr[], int n)

{

for (int i = 2; i < n; i++)

{

int j = i - 2;

int temp = arr[i];

if (((i + 1) & 1) == 1)

{

while (j >= 0 && temp >= arr[j])

{

arr[j + 2] = arr[j];

j -= 2;

}

arr[j + 2] = temp;

}

else

{

while (j >= 0 && temp <= arr[j])

{

arr[j + 2] = arr[j];

j -= 2;

}

arr[j + 2] = temp;

}

} }

static void printArray(int arr[], int n)

{

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

{

System.out.printf("%d ", arr[i]);

}

System.out.printf("\n");

}

public static void main(String[] args)

{

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

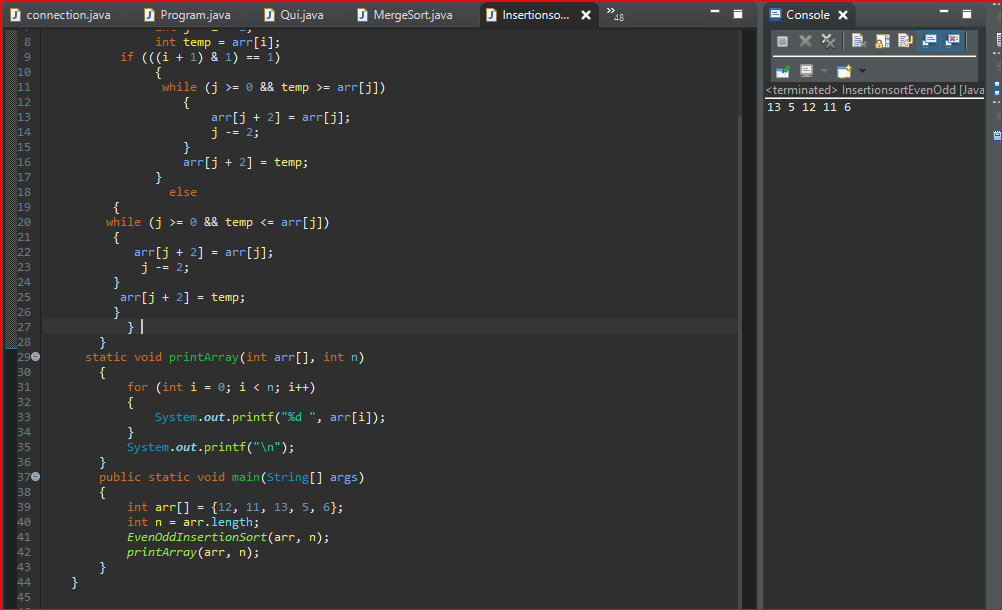
int n = arr.length;

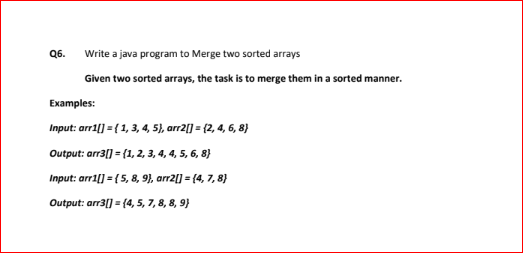
EvenOddInsertionSort(arr, n);

printArray(arr, n);

}

}





Soln:-

package hello;

class MergeSort

{

void merge(int arr[], int l, int m, int r)

{

int n1 = m - l + 1;

int n2 = r - m;

int L[] = new int [n1];

int R[] = new int [n2];

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

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

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

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

int i = 0, j = 0;

int k = l;

while (i < n1 && j < n2)

{

if (L[i] <= R[j])

{

arr[k] = L[i];

i++;

}

else

{

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1)

{

arr[k] = L[i];

i++;

k++;

}

while (j < n2)

{

arr[k] = R[j];

j++;

k++;

}

}

void sort(int arr[], int l, int r)

{ if (l < r)

{

int m = (l+r)/2;

sort(arr, l, m);

sort(arr , m+1, r);

merge(arr, l, m, r); }

}

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};

System.out.println("Given Array");

printArray(arr);

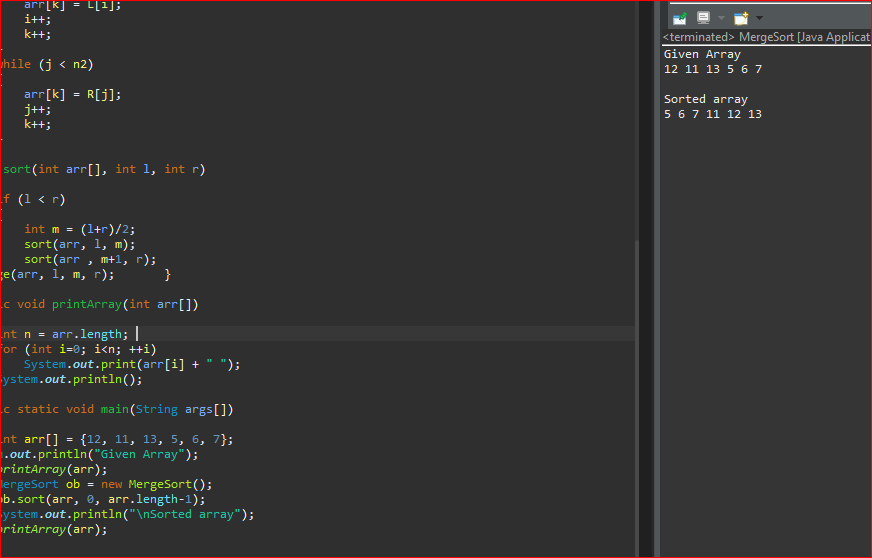
MergeSort ob = new MergeSort();

ob.sort(arr, 0, arr.length-1);

System.out.println("\nSorted array");

printArray(arr);

}}



->

public class Mergetwoarrays {

void sortArrays(int ar1[],int ar2[])

{

int len1 =ar1.length;

int len2 = ar2.length;

int len=len1+len2;

int aray[]=new int[len];

for(int i=0;i<len1;i++)

{

aray[i]=ar1[i] }

for(int j=0;j<len2;j++)

{

aray[j+4]=ar2[j];

}

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

{

int temp;

for(int j=i+1;j<len;j++)

{

if(aray[i]>aray[j])

{

temp=aray[i];

aray[i]=aray[j];

aray[j]=temp;

}

}

}

System.out.println("======Array after merging=======");

printArray(aray);

}

public static void printArray(int arr[])

{

System.out.print("{");

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

{

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

}

System.out.print("}");

System.out.println();

}

public static void main(String[] args) {

// TODO Auto-generated method stub

int ar1[]= {1,2,3,4};

int ar2[]= {3,4,5,6};

System.out.println("=====The first given array is==== ");

printArray(ar1);

System.out.println("======The second given array is==== ");

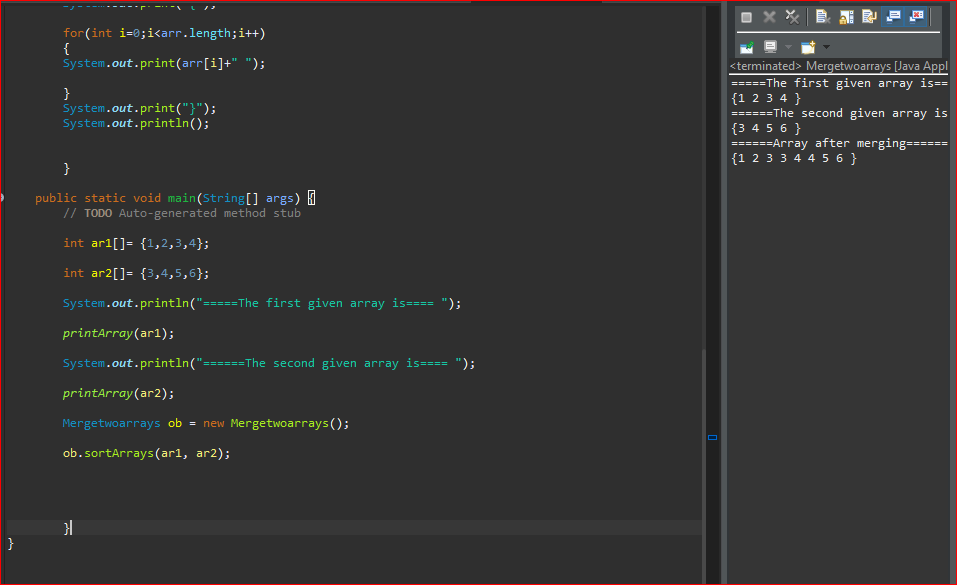
printArray(ar2);

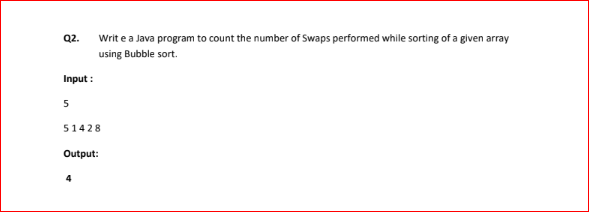
Mergetwoarrays ob = new Mergetwoarrays();

ob.sortArrays(ar1, ar2)

}

}





package hello;

public class Ques2 {

public static void swapcount(int arr[])

{

int count=0;

for(int i=0;i<arr.length-1;i++)

{

for(int j=0;j<arr.length-i-1;j++)

{

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

{

int temp=arr[j];

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

arr[j+1]=temp;

}

}

count++;

}

System.out.println("The number of swaps made == " + count);

}

public static void printarray(int ar[])

{

System.out.print("{");

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

{

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

}

System.out.print("}");

System.out.println();

}

public static void main(String args[])

{

int ar[]= {5,1,4,2,8};

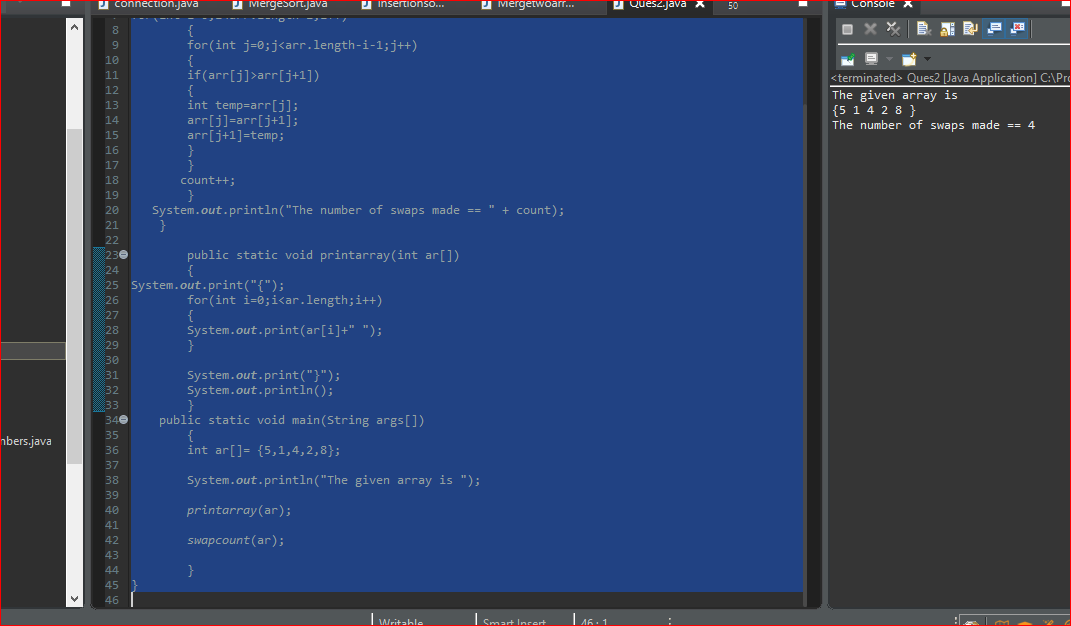
System.out.println("The given array is ");

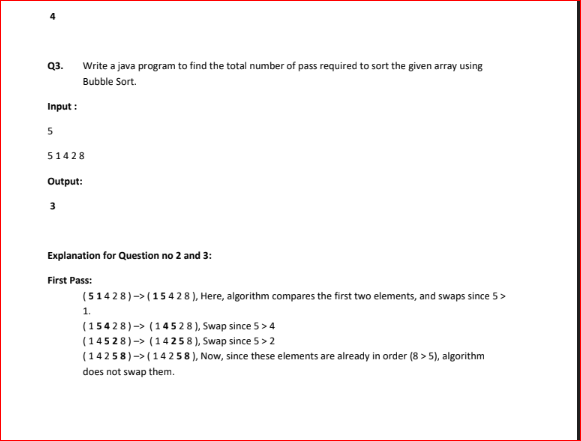
printarray(ar);

swapcount(ar);

}

}





public class Questn3 {

public static void passcount(int arr[])

{

int passcount=0;

for(int i=0;i<arr.length-1;i++)

{

int tes=0;

for(int j=0;j<arr.length-i-1;j++)

{

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

{

tes++;

int temp=arr[j];

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

arr[j+1]=temp;

}

}

if(tes!=0)

passcount++;

}

System.out.println("The number of swaps made == " + (passcount+1));

}

public static void printarray(int ar[])

{

System.out.print("{");

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

{

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

}

System.out.print("}");

System.out.println();

}

public static void main(String args[])

{

int ar[]= {5,1,4,2,8};

System.out.println("The given array is ");

printarray(ar);

passcount(ar);

}

}

