Q1. To merge two arrays of the same size sorted in descending order

ANS:-

Step 1:- Start.

Step 2:- Declare variable which takes the size of the array.

Step 3:- Declare the array A1 [], A2 [], A3 [].

Step 4:- Initialize the array.

Step 5:- While initializing the first array, copy the elements of the array to the merged array.

Step 6:- While initializing the second array, copy the elements of the array to the merged array.

Step 7:- Next step is to sort the merged array.

Step 8:- Display the resulting array.

Step 9:- Stop.

DRY RUN:-

N=3

A1 [3] = {3, 2, 1}

A2 [3] = {5, 6, 4}

A3 [6] = {6, 5, 4, 3, 2, 1}

Q2. To print all unique elements in an array.

ANS:-

Step 1:- Start.

Step 2:- Declare variable which takes the size of the array.

Step 3:- Declare the array A1 [].

Step 4:- Initialize the array.

Step 5:- Check if the current element found is occurring again in the array list.

Step 6:- If it is found, do not print that element.

Step 7:- Else print that element and continue.

Step 8:- Stop. DRY RUN:-

N=9

A1 [9] = {1, 3, 2, 1, 4, 4, 3, 6, 1, 8}

Ans [7] = {1, 2, 3, 4, 6, 8}

Q3. To separate odd and even integers in separate arrays.

ANS:-

Step 1:- Start.

Step 2:- Declare variable which takes the size of the array.

Step 3:- Declare the array A1 [].

Step 4:- Initialize the array.

Step 5:- Then create other two array A2 [], A3 [] which stores odd and even numbers respectively.

Step 6:- Check every elements of the given array whether it is divisible by 2 or not.

Step 7:- If it is divisible by 2 then store that element in A3 [] array and if not then store in a2 [] array.

Step 8:- Display odd and even array element.

Step 9:- Stop.

DRY RUN:-

N=10

A1 [10] = {1, 6, 2, 3, 4, 5, 7, 9, 10, 8}

A2 [5] = {1, 3, 5, 7, 9}

A3 [5] = {6, 2, 4, 10, 8}

Q4. To accept two matrices and check whether they are equal.

ANS:-

Step 1:- Start.

Step 2:- Declare the array A1 [] [], A2 [] [].

Step 3:- Initialize both the arrays.

Step 4:- Calculate the number of rows and columns in array A1 [] [] and store them in variables R1 and C1 respectively.

Step 5:- Calculate the number of rows and columns in array A2 [] [] and store them in variables R2 and C2 respectively.

Step 6:- Initialize the Flag to true.

Step 7:- Check if the size of both the arrays are same or not. If it is not same then display “Matrixes are not equal”.

Step 8:- If it is same then using for loop check all the elements of both the arrays.

Step 9:- If all the elements are same then display “Matrixes are equal”.

Step 10:- If any of the corresponding elements are not same then set the Flag to false and break the loop and display ”Matrixes are not equal”.

DRY RUN:-

A1 [] [] = {{1, 6, 2}, {3, 4, 5}, {7, 9, 10}}

A2 [] [] = = {{1, 6, 2}, {3, 4, 5}, {7, 9, 10}}

R1=3, C1=3, R2=3, C2=3

Flag=true;

”Matrixes are equal”

Q5. An array contains both positive and negative numbers in random order. Rearrange the array elements so that all negative numbers appear before all positive numbers.

ANS:-

Step 1:- Start.

Step 2:- Declare variable which takes the size of the array.

Step 3:- Declare the array A1 [].

Step 4:- Initialize the array.

Step 5:- Start traversing the array and if the current element is found to be negative then swap the current element with the first positive number and continue until all the elements are not encountered.

Step 6:- Print the rearranged elements of the array.

Step 7:- Stop.

DRY RUN:-

N=6

A1 [6] = {1, -1, 2, 2, 3,-3}

A1 [6] = {-1, -2, -3, 1, 3, 2}

Q6. Two sorted arrays, such that the arrays may have some common elements. Find the sum of the maximum sum path to reach from the beginning of any array to the end of any of the two arrays. We can switch from one array to another array only at common elements. And give the dry run for the same.

ANS:-

Step 1:- Start.

Step 2:- Declare the array A1 [], A2 [], variable Ans, Sum1, and Sum2;

Step 3:- Initialize the array.

Step 4:- Initialize the Ans=0, Sum1=0, Sum2=0. Here Sum1 and Sum2 are used to store sum of element in A1 [] and A2 [] respectively. These sums are between two common points.

Step 5:- Start traversing both the arrays. While traversing both the array compare the current element of both array A1 [] and A2 [].

Step 6:- If current element of A1 [] is smaller than current element of A2 [], then update Sum1, else if current element of A2 [] is smaller, then update Sum2.

Step 7:- If current element of A1 [] and A2 [] are same, then take the maximum of Sum1 and Sum2 and add it to the Ans. Also add the common element to the Ans.

Step 8:- Display the Ans.

Step 9:- Stop.

DRY RUN:-

A1 [] = {2, 3, 8, 10, 12}

A2 [] = {1, 5, 8, 9}

Ans=36

36 is sum of 1 + 5 + **8** + 10 + 12.

Q7. For given two sorted arrays and a number x, find the pair whose sum is closest to x and the pair has an element from each array

ANS:-

Step 1:- Start.

Step 2:- Declare the sorted array A1 [], A2 [], variable X, Diff and Result;

Step 3:- Initialize the arrays.

Step 4:- Initialize the X, Diff=INT\_MAX, Result=0.

Step 5:- Initialize two pointer left: = 0 and right: = n – 1.

Step 6:- While left <= A1.size () and right >= 0, do

If abs (A1 [left] + A2 [right] – x) < diff, then

Update diff and Result

If (A1 [left] + A2 [right]) < x, then

Increase left by 1

Otherwise

Decrease right by 1

Step 8:- Display the Result.

Step 9:- Stop.

DRY RUN:-

A1 [] = {1, 4, 5, 7}

A2 [] = {10, 20, 30, 40}

X=32

Result=1 and 30