Duplicate

#include <stdio.h>

int read\_array(int arr[], int size) {

printf("Enter Elements:\n");

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

if (scanf("%d", &arr[i]) != 1) {

printf("Invalid Input\n");

return 1;

}

}

return 0;

}

int count\_duplicates(int arr[], int size) {

int count = 0;

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

int freq = 0;

for (int j = 0; j < size; j++) {

if (arr[i] == arr[j]) {

freq++;

}

}

if (freq > 1) {

count++;

}

}

return count;

}

int main() {

int size;

printf("Enter Array Size:\n");

scanf("%d", &size);

if (size <= 0) {

printf("Invalid Input");

return 1;

}

int arr[size];

if (read\_array(arr, size) != 0) {

return 1;

}

int num\_duplicates = count\_duplicates(arr, size);

printf("No Of Duplicates: %d\n", num\_duplicates);

return 0;

}

2.

#include<stdio.h>

int removeDuplicates(int arr[], int size) {

int k = 0;

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

int isdupli = 0;

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

if(arr[i] == arr[j]) {

isdupli = 1;

break;

}

}

if(!isdupli) {

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

}

}

return k;

}

int main() {

int size;

scanf("%d", &size);

int arr[size];

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

scanf("%d", &arr[i]);

}

size = removeDuplicates(arr, size);

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

printf("%d\t", arr[i]);

}

return 0;

}

3.

#include <stdio.h>

int main() {

int size;

printf("Enter the size of the array: ");

scanf("%d", &size);

if (size < 1) {

printf("Array size must be at least 1.\n");

return 0;

}

int arr[size];

int unique[size]; // To store unique elements

int uniqueCount[size]; // To store the count of each unique element

int uniqueSize = 0; // To keep track of the number of unique elements

printf("Enter the elements of the array:\n");

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

scanf("%d", &arr[i]);

}

// Iterate over the array to find unique elements and their counts

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

int found = 0;

// Check if the element is already in the unique array

for (int j = 0; j < uniqueSize; j++) {

if (arr[i] == unique[j]) {

uniqueCount[j]++;

found = 1;

break;

}

}

// If the element is not in the unique array, add it

if (!found) {

unique[uniqueSize] = arr[i];

uniqueCount[uniqueSize] = 1;

uniqueSize++;

}

}

// Print the unique elements and their counts

printf("The array has %d unique elements:\n", uniqueSize);

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

printf("%d is repeated %d times\n", unique[i], uniqueCount[i]);

}

// Print the duplicates

printf("The duplicates in the array are:\n");

int hasDuplicates = 0; // Flag to check if there are any duplicates

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

if (uniqueCount[i] > 1) {

printf("%d\n", unique[i]);

hasDuplicates = 1;

}

}

if (!hasDuplicates) {

printf("No duplicates found.\n");

}

return 0;

}

4.

#include <stdio.h>

int main() {

int size;

printf("Enter the size of the array: ");

scanf("%d", &size);

if (size < 1) {

printf("Array size must be at least 1.\n");

return 0;

}

int arr[size];

printf("Enter the elements of the array:\n");

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

scanf("%d", &arr[i]);

}

// Bubble sort to sort the array

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

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

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

int temp = arr[j];

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

arr[j+1] = temp;

}

}

}

// Array to store unique elements (only once even if they are repeated)

int unique[size];

int uniqueSize = 0;

// Finding unique elements

int i = 0;

while (i < size) {

int count = 1;

while (i + 1 < size && arr[i] == arr[i+1]) {

count++;

i++;

}

// Add the element to the unique array (only once)

unique[uniqueSize++] = arr[i];

i++;

}

// Print the array with unique elements only

if (uniqueSize > 0) {

printf("The array with elements appearing only once is:\n");

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

printf("%d\n", unique[i]);

}

} else {

printf("No unique elements found.\n");

}

return 0;

}

5.

#include <stdio.h>

int main() {

int size;

// Input the size of the array

scanf("%d", &size);

int arr[size];

// Input the elements of the array

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

scanf("%d", &arr[i]);

}

// Bubble sort to sort the array

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

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

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

int temp = arr[j];

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

arr[j + 1] = temp;

}

}

}

int i = 0;

int totalDuplicates = 0;

// Check for duplicates and collect unique elements

int uniqueArr[size];

int uniqueIndex = 0;

while(i < size) {

int count = 1;

while(i < size - 1 && arr[i] == arr[i + 1]) {

count++;

i++;

}

if(count > 1) {

totalDuplicates += (count - 1); // Add number of duplicates

}

uniqueArr[uniqueIndex++] = arr[i]; // Add unique element to uniqueArr

i++;

}

// Output the total number of duplicates found

printf("Duplicates found:%d\n", totalDuplicates);

// Print the unique elements (duplicates removed)

printf("Array with duplicates removed: ");

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

printf("%d ", uniqueArr[i]);

}

printf("\n");

return 0;

}

Rotation

#include <stdio.h>

int getArrayInput(int arr[], int size);

void printArray(const int arr[], int size);

void rotateArray(int arr[], int size, int rotation);

int main() {

int size;

printf("Enter the array size: ");

if (scanf("%d", &size) != 1 || size <= 0) {

printf("Invalid Input\n");

return 1;

}

int arr[size];

if (!getArrayInput(arr, size)) {

printf("Invalid Input\n");

return 1;

}

int rotation;

printf("Enter the rotation value: ");

if (scanf("%d", &rotation) != 1) {

printf("Invalid Input\n");

return 1;

}

printf("Original array: ");

printArray(arr, size);

rotateArray(arr, size, rotation);

printf("Rotated array: ");

printArray(arr, size);

return 0;

}

// Function to get input for the array

int getArrayInput(int arr[], int size) {

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

printf("Enter element %d: ", i + 1);

if (scanf("%d", &arr[i]) != 1) {

return 0; // Indicate invalid input

}

}

return 1; // Indicate successful input

}

void printArray(const int arr[], int size) {

printf("[");

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

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

if (i < size - 1) {

printf(", ");

}

}

printf("]\n");

}

// Function to rotate the array

void rotateArray(int arr[], int size, int rotation) {

rotation %= size; // Handle rotations greater than array size

if (rotation == 0) {

return; // No rotation needed

}

int temp[size];

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

temp[i] = arr[(i + size - rotation) % size];

}

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

arr[i] = temp[i];

}

}

Min and max

#include <stdio.h>

int main() {

int size;

printf("Enter the size of the array: ");

scanf("%d", &size);

if (size <= 0) {

printf("Invalid Input\n");

return 1;

}

int arr[size];

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

printf("Enter element %d: ", i + 1);

if (scanf("%d", &arr[i]) != 1) {

printf("Invalid Input\n");

return 1;

}

}

int min = arr[0], max = arr[0];

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

if (arr[i] < min) {

min = arr[i];

} else if (arr[i] > max) {

max = arr[i];

}

}

printf("Minimum: %d\n", min);

printf("Maximum: %d\n", max);

return 0;

}

2.Second min(for second max iterate from back and change the condition according to it)

#include <stdio.h>

int main() {

int size;

// Input the size of the array

scanf("%d", &size);

int arr[size];

// Input the elements of the array

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

scanf("%d", &arr[i]);

}

// Bubble sort to sort the array

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

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

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

int temp = arr[j];

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

arr[j + 1] = temp;

}

}

}

int min=0;

for(int i=0;i<=size+1;i++){

if(arr[i]!=arr[size+1]){

min=arr[i];

break;

}

}

printf("%d",min);

}

Finding good pairs:

#include <stdio.h>

int main() {

int size;

// Get the size of the array

printf("Enter the size of the array: ");

if (scanf("%d", &size) != 1 || size <= 0) {

printf("Invalid input\n");

return 1;

}

int arr[size];

// Get the elements of the array

printf("Enter %d elements:\n", size);

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

if (scanf("%d", &arr[i]) != 1) {

printf("Invalid input\n");

return 1;

}

}

int sum;

// Get the target sum

printf("Enter the target sum: ");

if (scanf("%d", &sum) != 1) {

printf("Invalid input\n");

return 1;

}

int found = 0; // Flag to check if any pairs were found

// Find and print pairs that sum up to the target

printf("Pairs found:\n");

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

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

if (arr[i] + arr[j] == sum) {

printf("(%d, %d) ", arr[i], arr[j]);

found = 1;

}

}

}

// If no pairs were found

if (!found) {

printf("No pairs found\n");

}

printf("\n");

return 0;

}

Zerostoend

#include <stdio.h>

void moveZerosToEnd(int arr[], int size) {

int count = 0; // Count of non-zero elements

// Traverse the array. If an element is not zero, move it to the front

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

if (arr[i] != 0) {

arr[count++] = arr[i]; // Increment count only when a non-zero is found

}

}

// Fill remaining positions with zeros

while (count < size) {

arr[count++] = 0;

}

}

int main() {

int size = 8;

int arr[] = {0, 1, 9, 8, 4, 0, 2, 7};

printf("Original array: ");

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

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

}

printf("\n");

moveZerosToEnd(arr, size);

printf("Array after moving zeros to the end: ");

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

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

}

printf("\n");

return 0;

}

Think for front movement

2.

#include <stdio.h>

int getArrayInput(int arr[], int size);

void printArray(const int arr[], int size);

void zerosToStart(int arr[], int size);

int main() {

int size;

printf("Enter the array size: ");

if (scanf("%d", &size) != 1 || size <= 0) {

printf("Invalid Input\n");

return 1;

}

int arr[size];

if (!getArrayInput(arr, size)) {

printf("Invalid Input\n");

return 1;

}

zerosToStart(arr, size);

printArray(arr, size);

return 0;

}

// Function to get input for the array

int getArrayInput(int arr[], int size) {

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

printf("Enter element %d: ", i + 1);

if (scanf("%d", &arr[i]) != 1) {

return 0; // Indicate invalid input

}

}

return 1; // Indicate successful input

}

void printArray(const int arr[], int size) {

printf("[");

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

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

if (i < size - 1) {

printf(", ");

}

}

printf("]\n");

}

void zerosToStart(int arr[], int size) {

int zcount = size - 1; // Start from the end

int i;

for (i = size - 1; i >= 0; i--) {

if (arr[i] != 0) {

arr[zcount--] = arr[i];

}

}

// Fill the remaining positions with zeros

while (zcount >= 0) {

arr[zcount--] = 0;

}

}

Frequency

int found;

int count;

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

{

found = 0;

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

{

if(arr[j]==arr[i])

{

found = 1;

break;

}

}

if(!found)

{

count = 0;

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

{

if(arr[j]==arr[i])

{

count++;

}

}

printf("%d : %d \n",arr[i],count);

}

}

return 0;

}

2.

#include<stdio.h>

int main() {

int size;

scanf("%d", &size);

int arr[size];

// Input the array elements

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

scanf("%d", &arr[i]);

}

// Traverse through the array to find frequencies

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

if (arr[i] == -1) {

continue; // Skip already counted elements

}

int freq = 1; // Initial frequency of the current element

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

if (arr[i] == arr[j]) {

freq++;

arr[j] = -1; // Mark duplicate elements as counted

}

}

// Print the frequency of the current element

printf("%d occurs %d times\n", arr[i], freq);

}

return 0;

}

Delete element at specified position

#include <stdio.h>

int main() {

int size;

printf("Enter Array Size: ");

scanf("%d", &size);

if (size <= 0) {

printf("Invalid Input\n");

return 1;

}

int arr[size];

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

printf("Enter Element %d: ", i + 1);

if (scanf("%d", &arr[i]) != 1) {

printf("Invalid Input\n");

return 1;

}

}

int pos;

printf("Enter Position to Delete: ");

scanf("%d", &pos);

if (pos >= 0 && pos < size) {

for (int i = pos; i < size - 1; i++) {

arr[i] = arr[i + 1];

}

size--;

printf("Updated Array: ");

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

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

}

printf("\n");

} else {

printf("Invalid Position to Delete the Element\n");

}

return 0;

}

Product each

#include <stdio.h>

void computeProductArray(int arr[], int size, int result[]) {

int totalProduct = 1;

int zeroCount = 0;

// Compute total product and count zeros

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

if (arr[i] != 0) {

totalProduct \*= arr[i];

} else {

zeroCount++;

}

}

// Compute result based on zero count

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

if (zeroCount > 1) {

result[i] = 0;

} else if (zeroCount == 1) {

result[i] = (arr[i] == 0) ? totalProduct : 0;

} else {

result[i] = totalProduct / arr[i];

}

}

}

int main() {

int size;

printf("Enter the size of the array: ");

if (scanf("%d", &size) != 1 || size <= 0) {

printf("Invalid input\n");

return 1;

}

int arr[size];

int result[size];

printf("Enter the elements of the array: ");

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

if (scanf("%d", &arr[i]) != 1) {

printf("Invalid array elements\n");

return 1;

}

}

computeProductArray(arr, size, result);

printf("Output: [");

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

printf("%d", result[i]);

if (i < size - 1) {

printf(", ");

}

}

printf("]\n");

return 0;

}

#include<stdio.h>

int main() {

int size;

// Input the size of the array

scanf("%d", &size);

int arr[size];

// Input the elements of the array

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

scanf("%d", &arr[i]);

}

int productarr[size];

// Calculate the product of all elements except the current element

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

int product = 1;

for (int j = 0; j < size; j++) {

if (i != j) { // Skip the current element

product \*= arr[j];

}

}

productarr[i] = product;

}

// Print the resulting product array

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

printf("%d ", productarr[i]);

}

return 0;

}

3.

#include <stdio.h>

int main() {

int size;

scanf("%d", &size);

int arr[size];

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

scanf("%d", &arr[i]);

}

int proarr[size];

int prosize=0;

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

int pro=1;

for(int j=0;j<size;j++){

if(i==j){

continue;

}

pro\*=arr[j];

}

proarr[prosize++]=pro;

}

printf("[");

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

printf("%d",proarr[i]);

if(i<prosize-1){

printf(",");

}

}

printf("]");

}

Budget

#include <stdio.h>

#include <stdlib.h>

void findCombinations(int prices[], int n, int budget, int start, int current[], int currSize, int currentSum) {

if (currentSum == budget) {

printf("[");

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

printf("%d", current[i]);

if (i < currSize - 1) {

printf(", ");

}

}

printf("]\n");

return;

}

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

if (currentSum + prices[i] <= budget) {

current[currSize] = prices[i];

findCombinations(prices, n, budget, i + 1, current, currSize + 1, currentSum + prices[i]);

}

}

}

int main() {

int n, budget;

printf("Enter the number of prices: ");

if (scanf("%d", &n) != 1 || n < 0) {

printf("Invalid Input\n");

return 1;

}

if (n == 0) {

printf("Empty Array\n");

return 0;

}

int prices[n];

printf("Enter %d prices:\n", n);

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

if (scanf("%d", &prices[i]) != 1 || prices[i] < 0) {

printf("Invalid Input\n");

return 1;

}

}

printf("Enter the budget: ");

if (scanf("%d", &budget) != 1 || budget < 0) {

printf("Invalid Input\n");

return 1;

}

if (budget == 0) {

printf("Zero Budget\n");

return 0;

}

int current[n];

printf("Expected Output:\n");

findCombinations(prices, n, budget, 0, current, 0, 0);

return 0;

}

Removeduplicateson sorted arr

#include <stdio.h>

#include <stdbool.h>

// Function to remove duplicates from a sorted array

int removeDuplicates(int nums[], int n, int result[]) {

if (n == 0) return 0; // If the array is empty, return 0

int uniqueCount = 0; // Index to track unique elements

result[uniqueCount++] = nums[0]; // The first element is always unique

// Iterate through the array starting from the second element

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

// If the current element is different from the previous one, it's unique

if (nums[i] != nums[i - 1]) {

result[uniqueCount++] = nums[i]; // Add unique element to result array

}

}

return uniqueCount; // Return the number of unique elements

}

// Function to check if all array elements are within the valid range

bool isValidArray(int nums[], int n) {

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

// If any element is out of the range -100 to 100, return false

if (nums[i] < -100 || nums[i] > 100) {

return false;

}

}

return true; // All elements are within the valid range

}

int main() {

// Sample input array

int nums[] = {-10, -5, 0, 0, 3, 3, 5, 8};

int n = sizeof(nums) / sizeof(nums[0]); // Determine the number of elements

// Check if all elements are valid

if (!isValidArray(nums, n)) {

printf("Invalid array elements\n");

return 1; // Exit the program if invalid elements are found

}

int result[n]; // Array to store unique elements

int newLength = removeDuplicates(nums, n, result); // Remove duplicates and get new length

// Print the unique elements

printf("Output: [");

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

printf("%d", result[i]);

if (i < newLength - 1) {

printf(", ");

}

}

printf("]\n");

return 0; // Exit the program successfully

}

Symmetric matrix

#include<stdio.h>

int main(){

int r,c;

printf("Enter the dimensions:");

scanf("%d %d",&r,&c);

int mat[r][c];

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

for(int j=0;j<c;j++){

printf("Enter the elements mat[%d][%d]:",i,j);

if(scanf("%d",&mat[i][j])!=1){

printf("Invalid Input");

return 1;

}

}

}

int flag=1;

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

for(int j=0;j<c;j++){

if(mat[i][j]!=mat[j][i]){

flag=0;

break;

}

}

if(flag==0)break;

}

if(flag){

printf("Symmetric matrix");

}else{

printf("Not Symmetric matrix");

}

}

Extract digit

// Online C compiler to run C program online

#include <stdio.h>

int extractDigit(int arr[],int size,int result[],int \*resultSize){

int index=0;

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

int num=arr[i];

if(num<0){

printf("Invalid number");

}

if(num==0){

result[\*resultSize++]=0;

return 1;

}

int digit[10];

int count=0;

while(num>0){

digit[count++]=num%10;

num/=10;

}

for(int j=count-1;j>=0;j--){

result[index++]=digit[j];

}

}

\*resultSize=index;

}

int main() {

int size;

scanf("%d",&size);

int arr[size];

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

if(scanf("%d",&arr[i])!=1){

printf("Invalid Input");

return 1;

}

}

int resultSize;

int result[100];

extractDigit(arr,size,result,&resultSize);

printf("[");

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

printf("%d",result[i]);

if(i<resultSize-1){

printf(",");

}

}

printf("]");

}

Ones and twos complement

#include <stdio.h>

#include <string.h>

// Function to pad the binary number to 8 bits

void pad\_to\_8\_bits(char \*b) {

int len = strlen(b);

char padded[9] = "00000000";

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

padded[8 - len + i] = b[i];

}

strcpy(b, padded);

}

// Function to compute one's complement and two's complement

void compute\_complements(char \*b) {

char ones[9], twos[9];

int carry = 1;

for (int i = 0; i < 8; i++) {

ones[i] = (b[i] == '0') ? '1' : '0';

}

ones[8] = '\0';

for (int i = 7; i >= 0; i--) {

if (ones[i] == '1' && carry == 1) {

twos[i] = '0';

} else {

twos[i] = ones[i] + carry;

carry = 0;

}

}

twos[8] = '\0';

printf("One's Complement: %s\n", ones);

printf("Two's Complement: %s\n", twos);

}

// Function to validate if the input is a binary number

int is\_valid\_binary(char \*b) {

for (int i = 0; i < strlen(b); i++) {

if (b[i] != '0' && b[i] != '1') {

return 0;

}

}

return 1;

}

int main() {

char b[9];

printf("Enter a binary number: ");

scanf("%8s", b);

if (!is\_valid\_binary(b)) {

printf("Invalid binary number\n");

return 1;

}

// Pad the binary number to 8 bits

pad\_to\_8\_bits(b);

printf("Input Binary: %s\n", b);

// Compute one's and two's complement

compute\_complements(b);

return 0;

}

Leader number

#include<stdio.h>

void findLeaders(int arr[], int size) {

int maxFromRight = arr[size - 1]; // The rightmost element is always a leader

printf("%d ", maxFromRight); // Print the first leader (rightmost element)

// Traverse the array from right to left

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

if (arr[i] > maxFromRight) {

maxFromRight = arr[i]; // Update maxFromRight

printf("%d ", maxFromRight); // Print the leader

}

}

printf("\n");

}

int main() {

int size;

// Input the size of the array

printf("Enter the size of the array: ");

scanf("%d", &size);

int arr[size];

// Input the elements of the array

printf("Enter the elements of the array:\n");

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

scanf("%d", &arr[i]);

}

// Find and print all leader elements

printf("Leader elements in the array are:\n");

findLeaders(arr, size);

return 0;

}

Even and odd

#include <stdio.h>

int main() {

int size;

// Input the size of the array

scanf("%d", &size);

int arr[size];

// Input the elements of the array

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

scanf("%d", &arr[i]);

}

// Bubble sort to sort the array in ascending order

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

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

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

int temp = arr[j];

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

arr[j + 1] = temp;

}

}

}

// Print even numbers first

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

if(arr[i] % 2 == 0) {

printf("%d\t", arr[i]);

}

}

printf("\n");

// Print odd numbers next

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

if(arr[i] % 2 != 0) {

printf("%d\t", arr[i]);

}

}

return 0;

}

Unique element

#include<stdio.h>

// Function to remove duplicates in a sorted array

int removeDuplicates(int arr[], int size) {

if (size == 0) return 0; // Handle edge case of empty array

int j = 0; // Index to keep track of the last unique element

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

if (arr[j] != arr[i]) { // If the current element is different from the last unique element

j++;

arr[j] = arr[i]; // Move the unique element to the next position

}

}

return j + 1; // Return the new size of the array

}

int main() {

int size;

// Input the size of the array

printf("Enter the size of the array: ");

scanf("%d", &size);

int arr[size];

// Input the elements of the array

printf("Enter the elements of the array:\n");

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

scanf("%d", &arr[i]);

}

// Sort the array using bubble sort

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

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

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

int temp = arr[j];

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

arr[j + 1] = temp;

}

}

}

// Remove duplicates from the sorted array

size = removeDuplicates(arr, size);

// Print the resulting array

printf("Array after removing duplicates:\n");

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

printf("%d\t", arr[i]);

}

return 0;

}

Merge

#include <stdio.h>

int main() {

int size1;

scanf("%d", &size1);

int arr1[size1];

// Input the elements for the first array

for(int i = 0; i < size1; i++) {

scanf("%d", &arr1[i]);

}

int size2;

scanf("%d", &size2);

int arr2[size2];

// Input the elements for the second array

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

scanf("%d", &arr2[i]);

}

int n = size1 + size2;

int armour[n];

// Copy elements from the first array to the armour array

for(int i = 0; i < size1; i++) {

armour[i] = arr1[i];

}

// Copy elements from the second array to the armour array

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

armour[size1 + i] = arr2[i];

}

// Print the merged array

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

printf("%d\t", armour[i]);

}

return 0;

}

Range finder

#include <stdio.h>

void printRange(int start, int end) {

if (start == end) {

printf("%d", start);

} else {

printf("%d-%d", start, end);

}

}

int main() {

int size;

scanf("%d", &size);

int arr[size];

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

scanf("%d", &arr[i]);

}

// Iterate through the array to find consecutive ranges

int start=arr[0];

int end=arr[0];

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

if(arr[i]==end+1){

end=arr[i];

}else{

printRange(start,end);

printf(",");

start=arr[i];

end=arr[i];

}

}

printRange(start,end);

return 0;

}

Fibonacci series in arr

#include <stdio.h>

int main() {

int size;

// Get the number of terms to be stored in the array

printf("Enter the number of terms to be stored in the fib array: ");

if (scanf("%d", &size) != 1 || size <= 0) {

printf("Invalid input\n");

return 1;

}

int fib[size];

int count = 0;

int a = 1, b = 2; // Initialize the first two Fibonacci numbers

// Store 1 as the first term if required

if (size > 0) {

fib[count++] = 1;

}

// Generate and store even Fibonacci numbers

while (count < size) {

if (b % 2 == 0) { // Check if the Fibonacci number is even

fib[count++] = b;

}

int next = a + b; // Generate the next Fibonacci number

a = b;

b = next;

}

// Print the fib array

printf("fib[] = [");

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

printf("%d", fib[i]);

if (i < size - 1) {

printf(", ");

}

}

printf("]\n");

return 0;

}

Input

n=8

Output

3

Explanation:

1

23

456

78(incomplete)