Assignment – 2 (Array Concepts)

1 An array is a data structure containing a collection of values or variables. The simplest type of array is a linear array or one-dimensional array. An array can be defined in C with the following syntax:

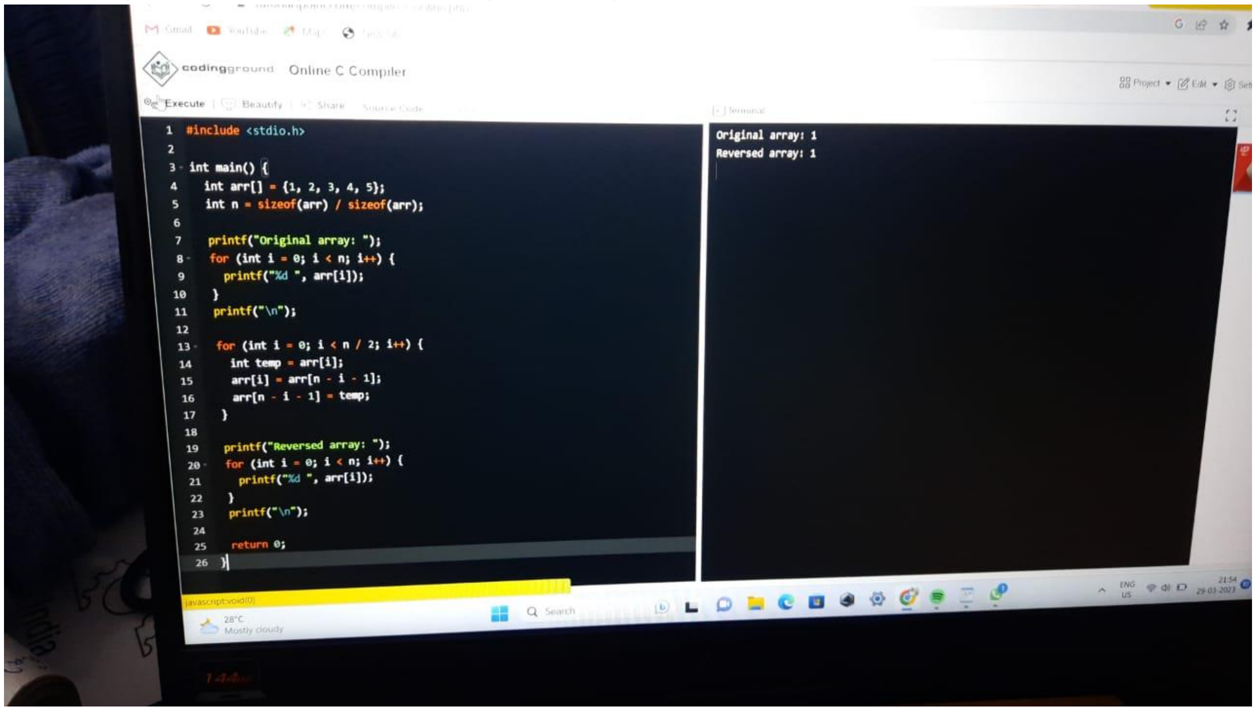
int Arr[5] = {12, 56, 34, 78, 100};

/* here 12,56,34,78,100 are the elements at indices 0,1,2,3,4 respectively */

In this example, array Arr is a collection of 5 integers. Each integer can be identified and accessed by its index. The indices of the array start with 0, so the first element of the array will have index 0, the next will have index 1 and so on.

Largest element of the array is the array element which has the largest numerical value among all the array elements.

Examples:
If we are entering 5 elements (N = 5), with array element values as 12, 56, 34, 78 and 100
Then, largest element present in the given array is: 100



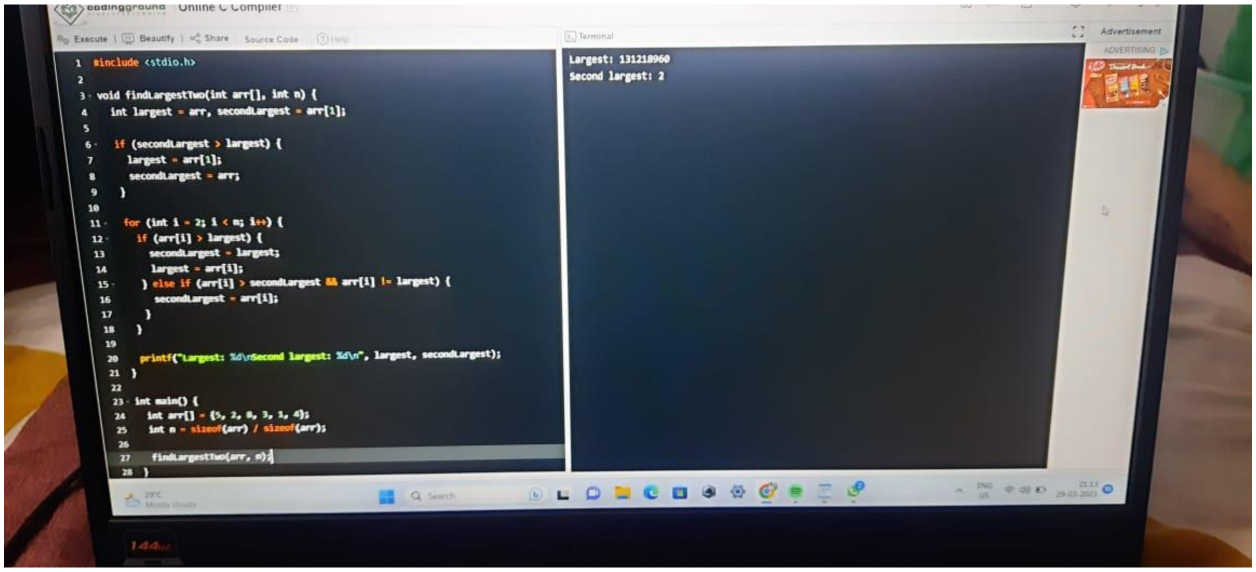
```c
#include <stdio.h>

int main() {
  int arr[] = {1, 2, 3, 4, 5};
  int n = sizeof(arr) / sizeof(arr);

  printf("Original array: ");
  for (int i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }
  printf("\n");

  for (int i = 0; i < n / 2; i++) {
    int temp = arr[i];
    arr[i] = arr[n - i - 1];
    arr[n - i - 1] = temp;
  }

  printf("Reversed array: ");
  for (int i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }
  printf("\n");

  return 0;
}
```

```
Original array: 1
Reversed array: 1
```

2 Problem Description

We have to write a program in C such that the program will read the elements of a one-dimensional array, then compares the elements and finds which are the largest two elements in a given array.
Expected Input and Output
1. Finding Largest 2 numbers in an array with unique elements:
If we are entering 5 elements (N = 5), with array element values as 2,4,5,8 and 7 then,
The FIRST LARGEST = 8
THE SECOND LARGEST = 7

2. Finding Largest 2 numbers in an array with recurring elements:
If we are entering 6 elements (N = 6), with array element values as 2,1,1,2,1 and 2 then,
The FIRST LARGEST = 2
THE SECOND LARGEST = 1

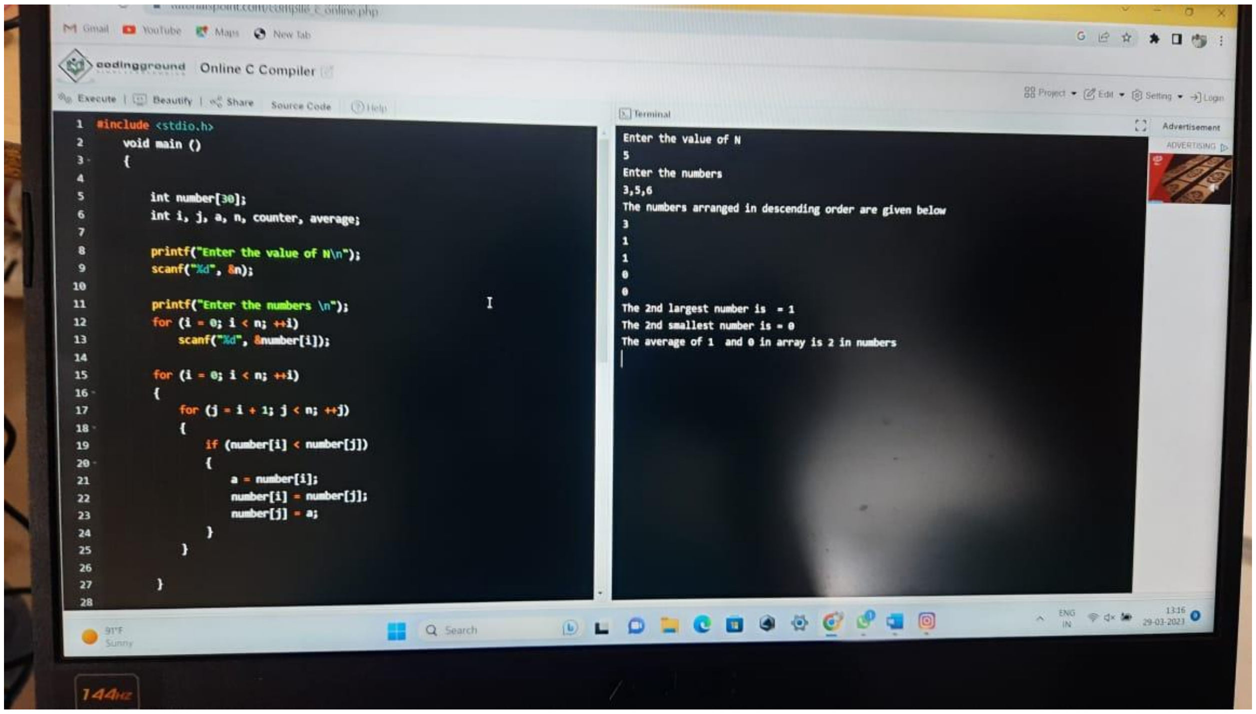

codingground Online C Compiler
Execute | Beautify | Share Source Code Help
Terminal
Advertisement

```c
#include <stdio.h>

void findLargestTwo(int arr[], int n) {
  int largest = arr, secondLargest = arr[1];

  if (secondLargest > largest) {
    largest = arr[1];
    secondLargest = arr;
  }

  for (int i = 2; i < n; i++) {
    if (arr[i] > largest) {
      secondLargest = largest;
      largest = arr[i];
    } else if (arr[i] > secondLargest && arr[i] != largest) {
      secondLargest = arr[i];
    }
  }

  printf("Largest: %d\nSecond largest: %d\n", largest, secondLargest);
}

int main() {
  int arr[] = {5, 2, 8, 3, 1, 4};
  int n = sizeof(arr) / sizeof(arr);

  findLargestTwo(arr, n);
}
```

```
Largest: 131218960
Second largest: 2
```

3 C Program finds second largest & smallest elements in an Array.

Problem Description
The program will implement a one dimensional array and sort the array in descending order. Then it finds the second largest and smallest element in an array and also find the average of these two array elements. Later it checks if the resultant average number is present in a given array. If found, display appropriate message.

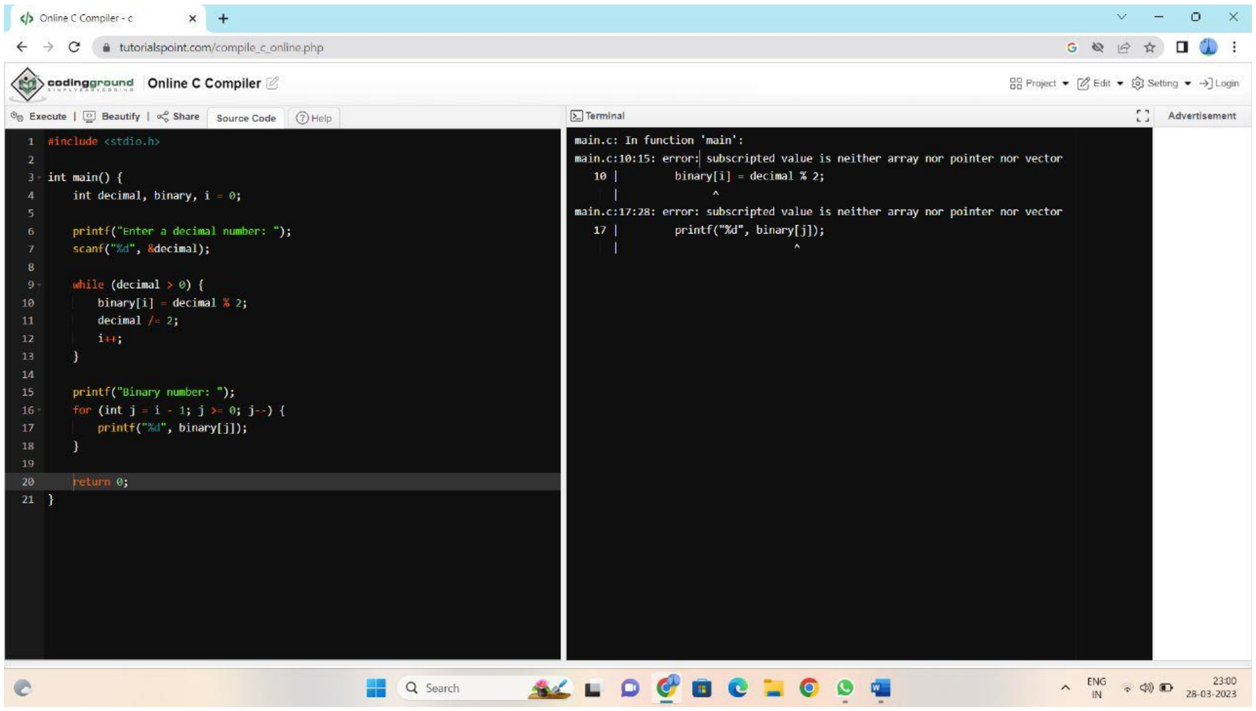


```c
#include <stdio.h>
    void main ()
    {

        int number[30];
        int i, j, a, n, counter, average;

        printf("Enter the value of N\n");
        scanf("%d", &n);

        printf("Enter the numbers \n");
        for (i = 0; i < n; ++i)
            scanf("%d", &number[i]);

        for (i = 0; i < n; ++i)
        {
            for (j = i + 1; j < n; ++j)
            {
                if (number[i] < number[j])
                {
                    a = number[i];
                    number[i] = number[j];
                    number[j] = a;
                }
            }

        }

```

```
Enter the value of N
5
Enter the numbers
3,5,6
The numbers arranged in descending order are given below
3
1
1
0
0
The 2nd largest number is  = 1
The 2nd smallest number is = 0
The average of 1  and 0 in array is 2 in numbers
```

4 C Program To Find Maximum Difference Between Two Elements in an Array
Example:

Consider the Following Array
int array[] = {10, 15, 90, 200, 110};
Output:
Maximum difference is 190
That is 200-10=190



```c
1  #include <stdio.h>
2
3  int main() {
4      int decimal, binary, i = 0;
5
6      printf("Enter a decimal number: ");
7      scanf("%d", &decimal);
8
9      while (decimal > 0) {
10         binary[i] = decimal % 2;
11         decimal /= 2;
12         i++;
13     }
14
15     printf("Binary number: ");
16     for (int j = i - 1; j >= 0; j--) {
17         printf("%d", binary[j]);
18     }
19
20     return 0;
21 }
```

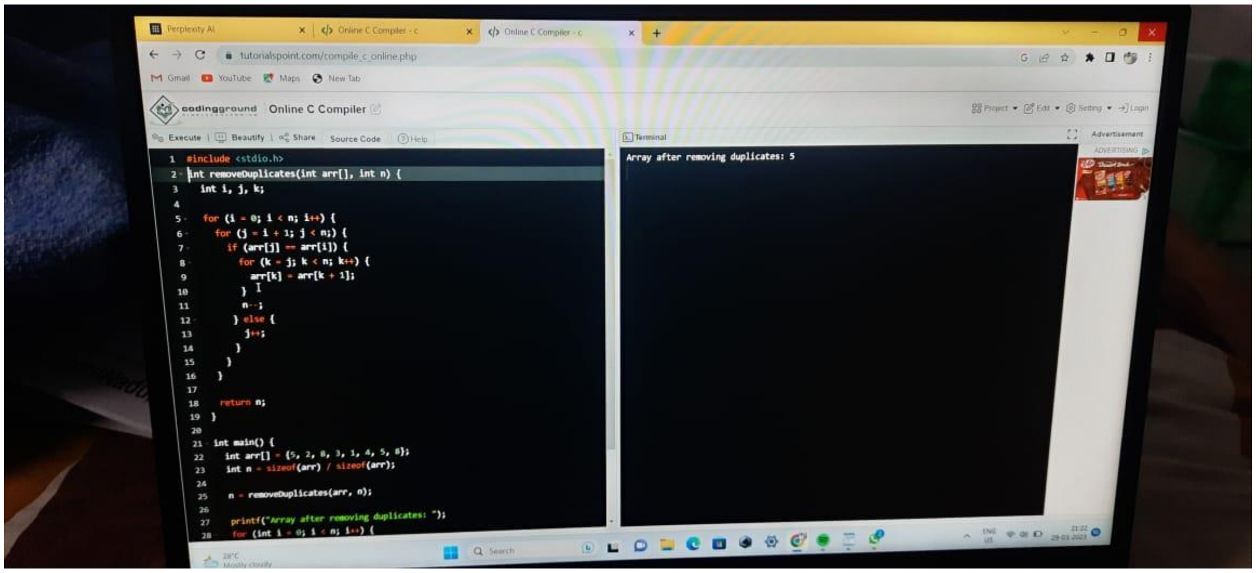
```
main.c: In function 'main':
main.c:10:15: error: subscripted value is neither array nor pointer nor vector
   10 |         binary[i] = decimal % 2;
      |               ^
main.c:17:28: error: subscripted value is neither array nor pointer nor vector
   17 |         printf("%d", binary[j]);
      |                            ^
```

5 C program to remove duplicate elements in an Array?
An array is a collection of similar data elements stored in a contiguous memory location.

Example: arr[5] = {2,7,1,23,5}

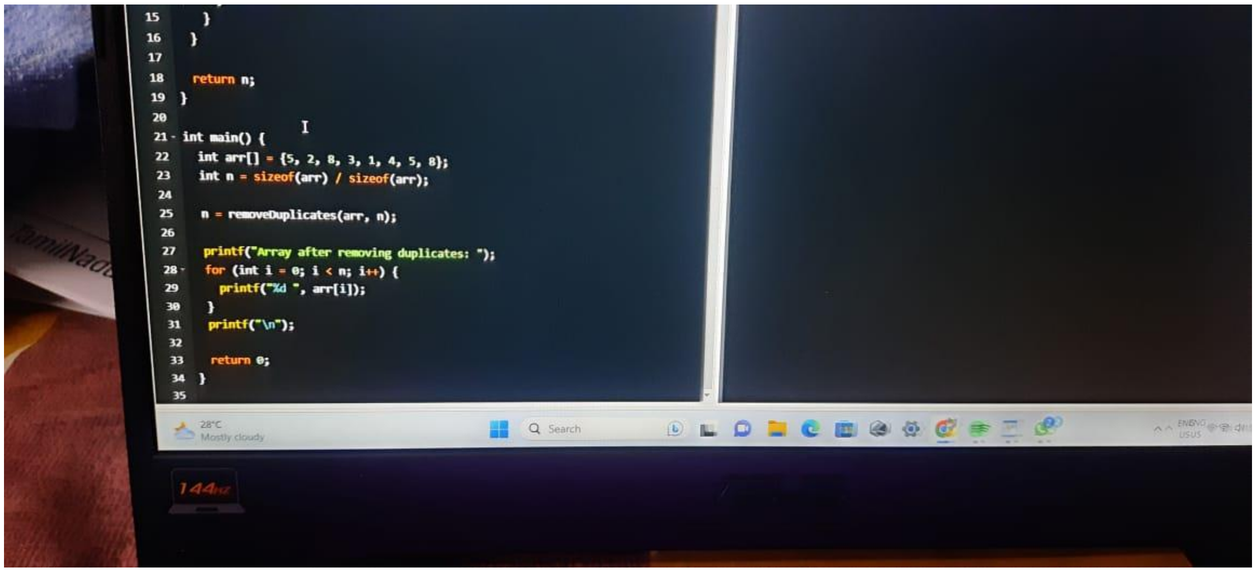
Example:

Input Array: 1,2,4,5,4,2,7,5
Output: Resultant Array after removing duplicates: 1,2,4,5,7



```c
#include <stdio.h>
int removeDuplicates(int arr[], int n) {
  int i, j, k;

  for (i = 0; i < n; i++) {
    for (j = i + 1; j < n;) {
      if (arr[j] == arr[i]) {
        for (k = j; k < n; k++) {
          arr[k] = arr[k + 1];
        }
        n--;
      } else {
        j++;
      }
    }
  }

  return n;
}

int main() {
  int arr[] = {5, 2, 8, 3, 1, 4, 5, 8};
  int n = sizeof(arr) / sizeof(arr);

  n = removeDuplicates(arr, n);

  printf("Array after removing duplicates: ");
  for (int i = 0; i < n; i++) {
```

```
Array after removing duplicates: 5
```

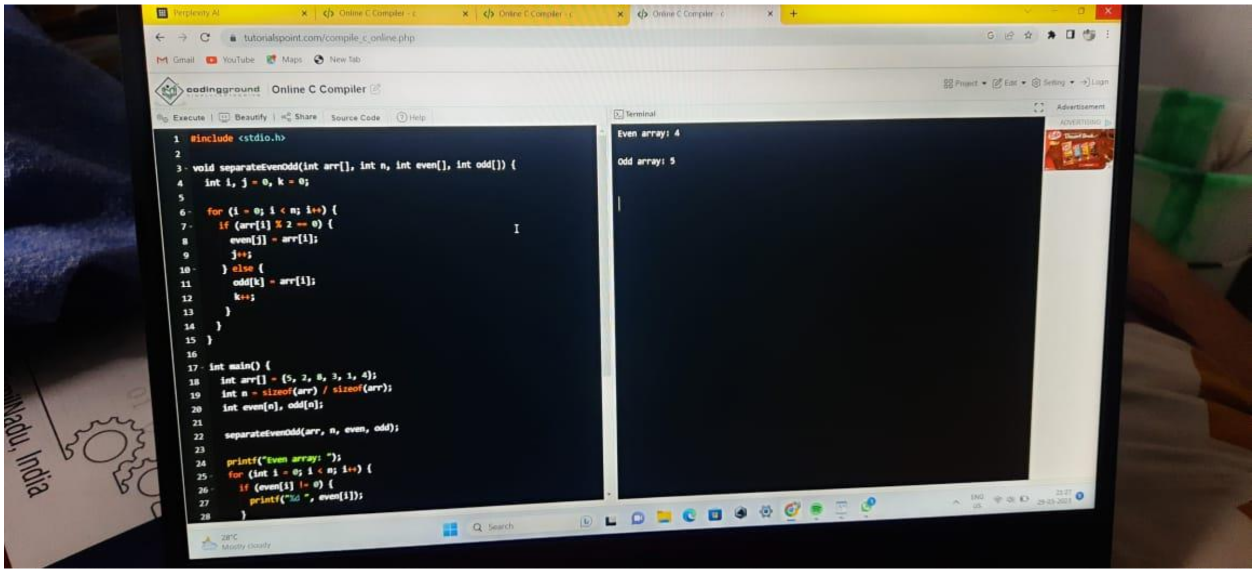


```c
    }
  }

  return n;
}

int main() {
  int arr[] = {5, 2, 8, 3, 1, 4, 5, 8};
  int n = sizeof(arr) / sizeof(arr);

  n = removeDuplicates(arr, n);

  printf("Array after removing duplicates: ");
  for (int i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }
  printf("\n");

  return 0;
}
```

6 C Program to put even & odd elements of an array in 2 separate arrays.

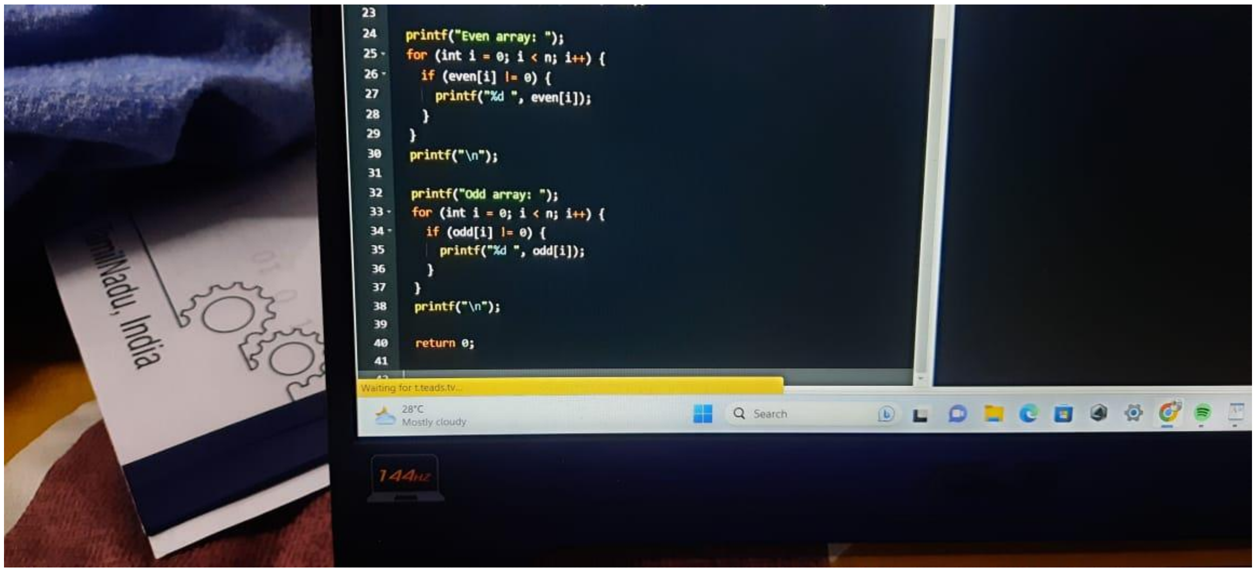
Problem Description

The program first finds the odd and even elements of the array. Then the odd elements of an array is stored in one array and even elements of an array is stored in another array.



```c
#include <stdio.h>

void separateEvenOdd(int arr[], int n, int even[], int odd[]) {
  int i, j = 0, k = 0;

  for (i = 0; i < n; i++) {
    if (arr[i] % 2 == 0) {
      even[j] = arr[i];
      j++;
    } else {
      odd[k] = arr[i];
      k++;
    }
  }
}

int main() {
  int arr[] = {5, 2, 8, 3, 1, 4};
  int n = sizeof(arr) / sizeof(arr);
  int even[n], odd[n];

  separateEvenOdd(arr, n, even, odd);

  printf("Even array: ");
  for (int i = 0; i < n; i++) {
    if (even[i] != 0) {
      printf("%d ", even[i]);
    }
```

```
Even array: 4

Odd array: 5
```

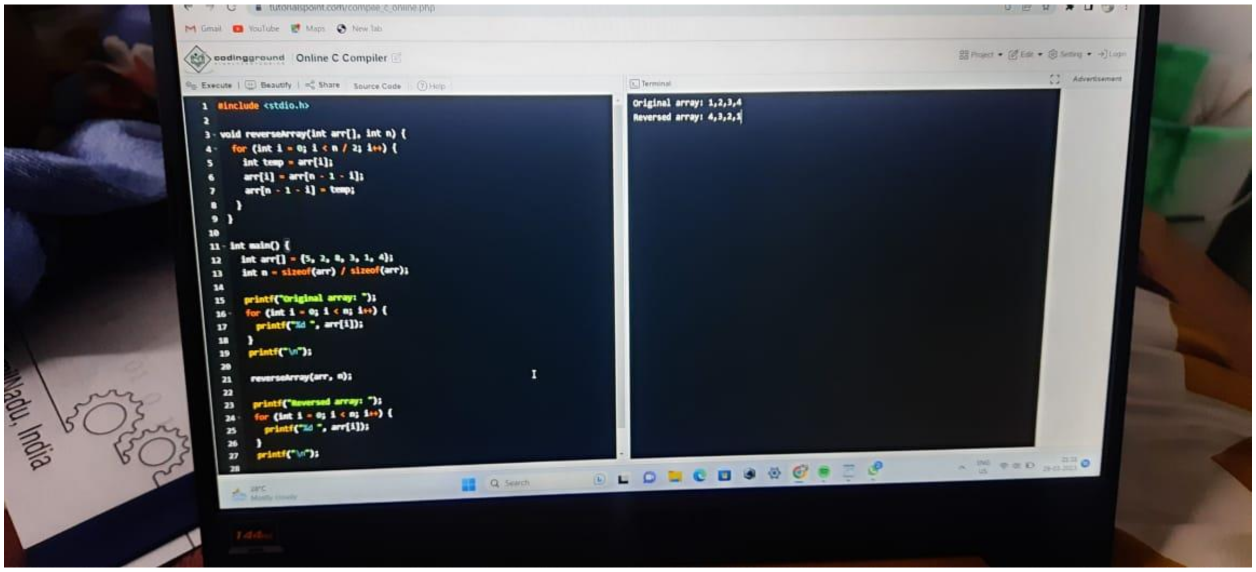


```c
  printf("Even array: ");
  for (int i = 0; i < n; i++) {
    if (even[i] != 0) {
      printf("%d ", even[i]);
    }
  }
  printf("\n");

  printf("Odd array: ");
  for (int i = 0; i < n; i++) {
    if (odd[i] != 0) {
      printf("%d ", odd[i]);
    }
  }
  printf("\n");

  return 0;
```

7 Reversing an array means substituting the last element in the first position and vice versa and doing such a thing for all elements of the array. For example, first element is swapped with last, second element is swapped by second last and so on.
Such arrays where the original and reversed arrays are equal are called palindrome arrays.
Examples:
Input array: [1,2,3,4]

Reversed array: [4,3,2,1]

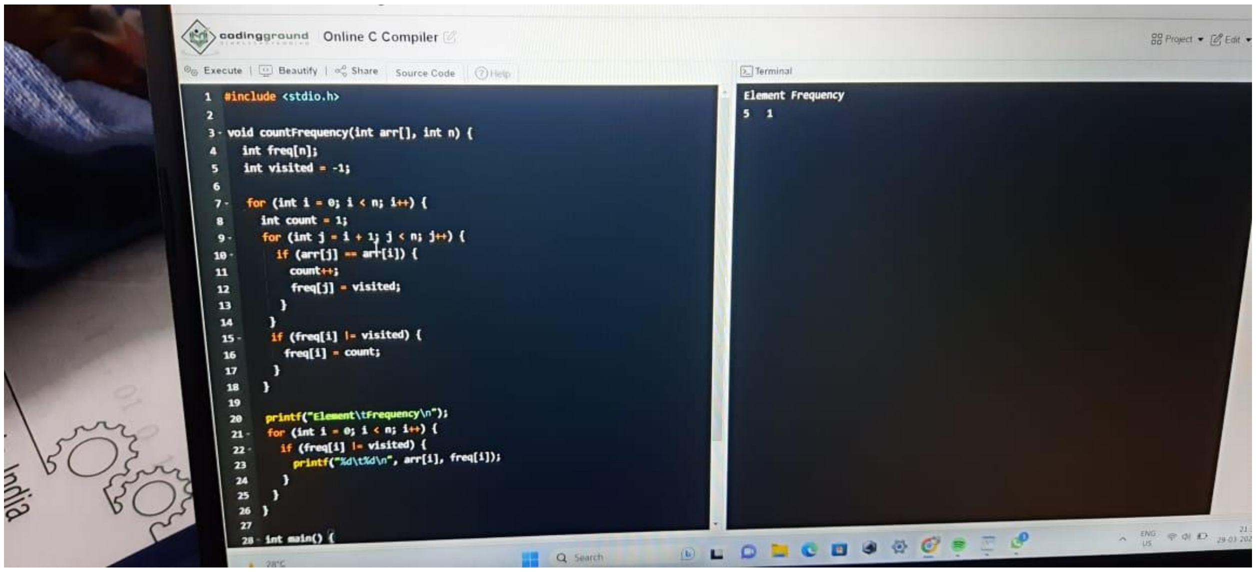
Input array: [3,2,1]
Reversed array: [1,2,3]



```c
#include <stdio.h>

void reverseArray(int arr[], int n) {
  for (int i = 0; i < n / 2; i++) {
    int temp = arr[i];
    arr[i] = arr[n - 1 - i];
    arr[n - 1 - i] = temp;
  }
}

int main() {
  int arr[] = {5, 2, 8, 3, 1, 4};
  int n = sizeof(arr) / sizeof(arr);

  printf("Original array: ");
  for (int i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }
  printf("\n");

  reverseArray(arr, n);

  printf("Reversed array: ");
  for (int i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }
  printf("\n");
```

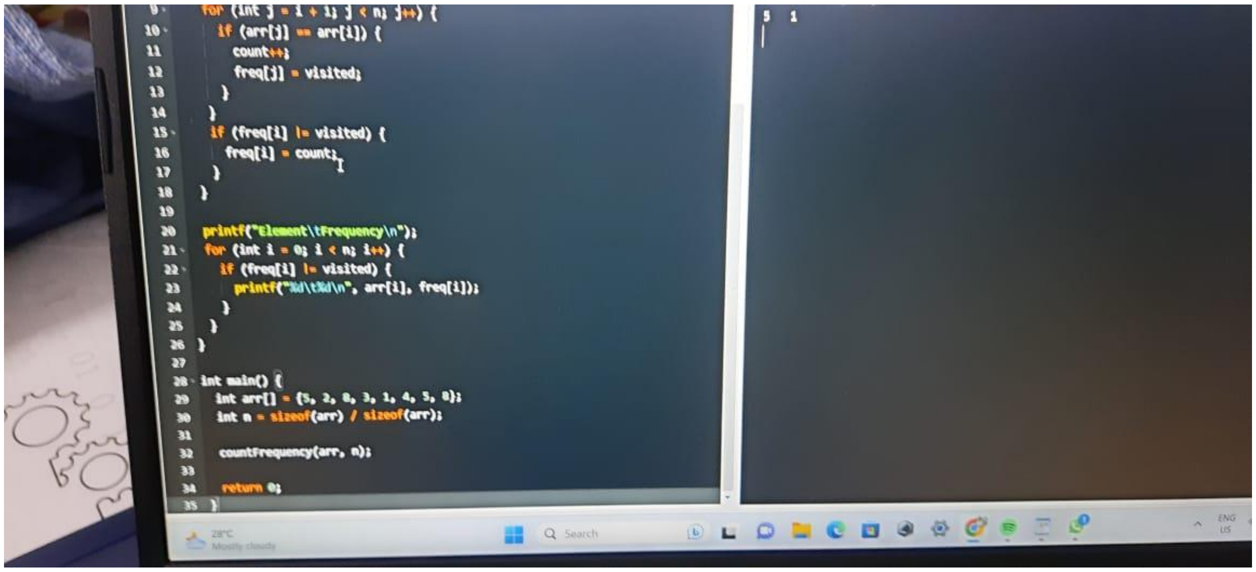
```
Original array: 1,2,3,4
Reversed array: 4,3,2,1
```

8 Write a program in C to count the frequency of each element of an array.



```c
#include <stdio.h>

void countFrequency(int arr[], int n) {
  int freq[n];
  int visited = -1;

  for (int i = 0; i < n; i++) {
    int count = 1;
    for (int j = i + 1; j < n; j++) {
      if (arr[j] == arr[i]) {
        count++;
        freq[j] = visited;
      }
    }
    if (freq[i] != visited) {
      freq[i] = count;
    }
  }

  printf("Element\tFrequency\n");
  for (int i = 0; i < n; i++) {
    if (freq[i] != visited) {
      printf("%d\t%d\n", arr[i], freq[i]);
    }
  }
}

int main() {
```

```
Element Frequency
5   1
```



```c
    for (int j = i + 1; j < n; j++) {
      if (arr[j] == arr[i]) {
        count++;
        freq[j] = visited;
      }
    }
    if (freq[i] != visited) {
      freq[i] = count;
    }
  }

  printf("Element\tFrequency\n");
  for (int i = 0; i < n; i++) {
    if (freq[i] != visited) {
      printf("%d\t%d\n", arr[i], freq[i]);
    }
  }
}

int main() {
  int arr[] = {5, 2, 8, 3, 1, 4, 5, 8};
  int n = sizeof(arr) / sizeof(arr);

  countFrequency(arr, n);

  return 0;
}
```

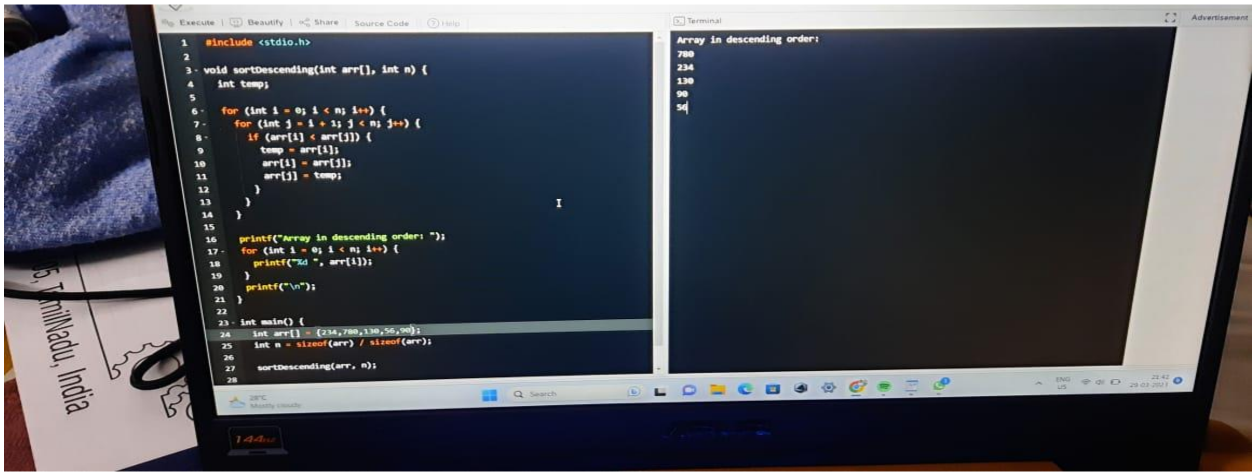
```
5   1
```

9 C Program to sort an array in descending order.

Problem Description

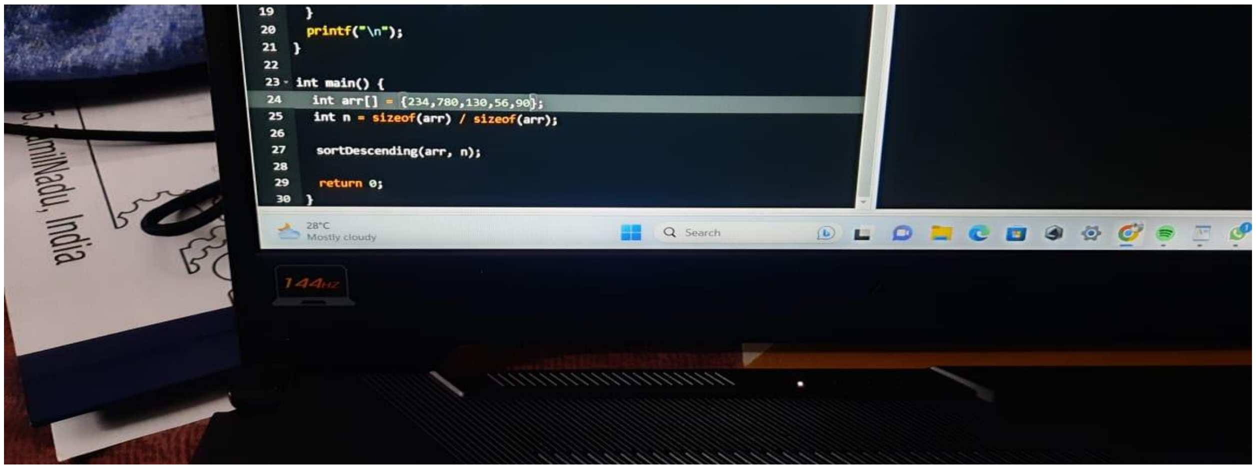
This program will implement a one-dimensional array of some fixed size, filled with some random numbers, then will sort all the filled elements of the array.

```
Enter the value of N
5
Enter the numbers
234
780
130
56
90
The numbers arranged in descending order are given below
780
234
130
90
56
```



```c
#include <stdio.h>

void sortDescending(int arr[], int n) {
  int temp;

  for (int i = 0; i < n; i++) {
    for (int j = i + 1; j < n; j++) {
      if (arr[i] < arr[j]) {
        temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
      }
    }
  }

  printf("Array in descending order: ");
  for (int i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }
  printf("\n");
}

int main() {
  int arr[] = {234,780,130,56,90};
  int n = sizeof(arr) / sizeof(arr);

  sortDescending(arr, n);

```

```
Array in descending order:
780
234
130
90
56
```



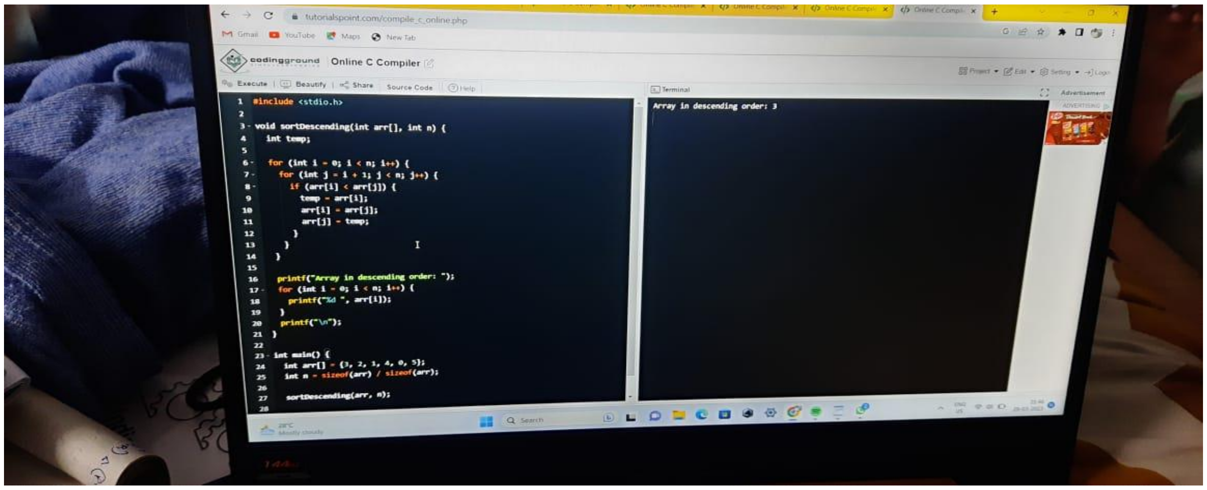
```c
  }
  printf("\n");
}

int main() {
  int arr[] = {234,780,130,56,90};
  int n = sizeof(arr) / sizeof(arr);

  sortDescending(arr, n);

  return 0;
}
```

10 Given an array arr[] where each element represents the max number of steps that can be made forward from that index. The task is to find the minimum number of jumps to reach the end of the array starting from index 0. If the end isn't reachable, return -1.

Examples:

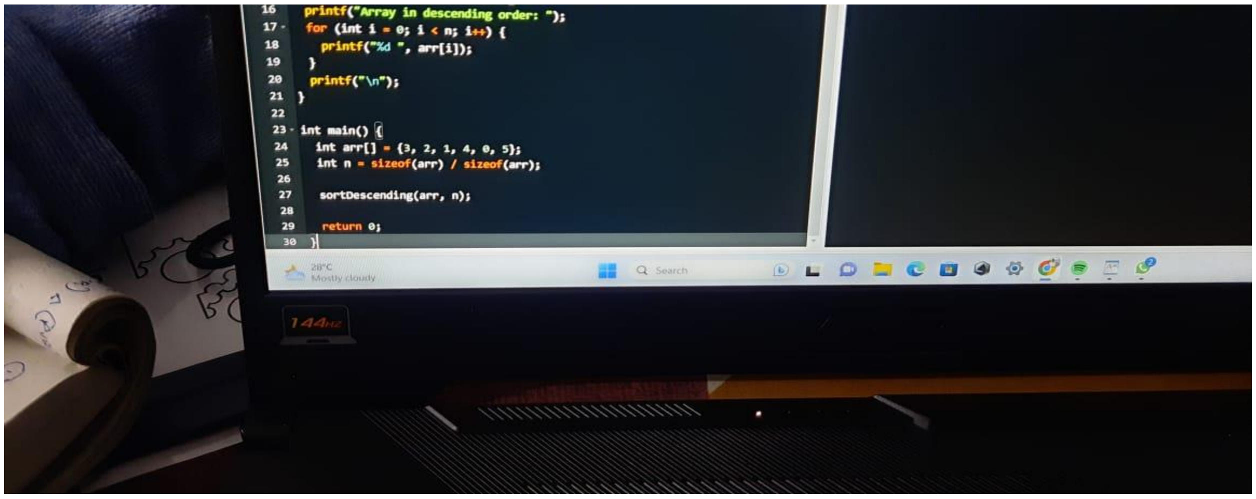
Input: arr[] = {1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9}
Output: 3 (1-> 3 -> 9 -> 9)
Explanation: Jump from 1st element to 2nd element as there is only 1 step.
Now there are three options 5, 8 or 9. I
f 8 or 9 is chosen then the end node 9 can be reached. So 3 jumps are made.

Input: arr[] = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
Output: 10
Explanation: In every step a jump is needed so the count of jumps is 10.



```c
#include <stdio.h>

void sortDescending(int arr[], int n) {
  int temp;

  for (int i = 0; i < n; i++) {
    for (int j = i + 1; j < n; j++) {
      if (arr[i] < arr[j]) {
        temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
      }
    }
  }

  printf("Array in descending order: ");
  for (int i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }
  printf("\n");
}

int main() {
  int arr[] = {3, 2, 1, 4, 0, 5};
  int n = sizeof(arr) / sizeof(arr);

  sortDescending(arr, n);

```

```
Array in descending order: 3
```



```c
  printf("Array in descending order: ");
  for (int i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }
  printf("\n");
}

int main() {
  int arr[] = {3, 2, 1, 4, 0, 5};
  int n = sizeof(arr) / sizeof(arr);

  sortDescending(arr, n);

  return 0;
}
```