

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 */

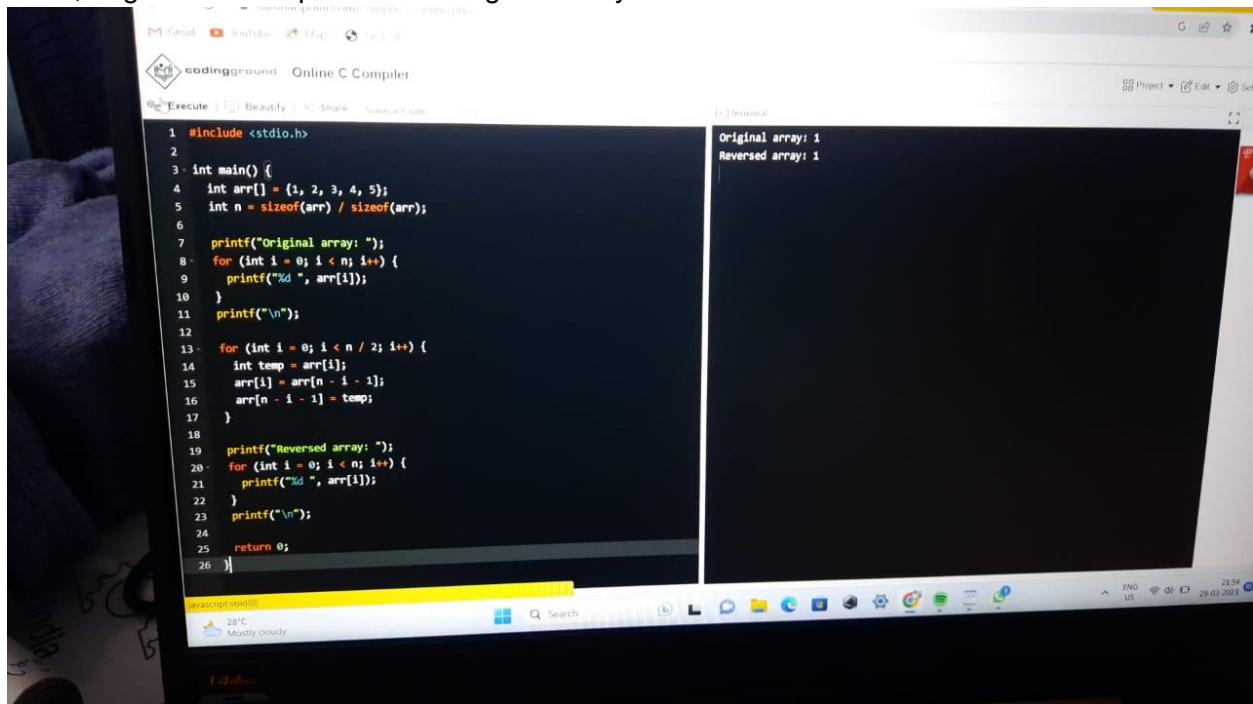
In this example, array Arr is a collection of 5 integers. Each integer can be identified and accessed by its index. The indices of the array start with 0, so the first element of the array will have index 0, the next will have index 1 and so on.

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



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

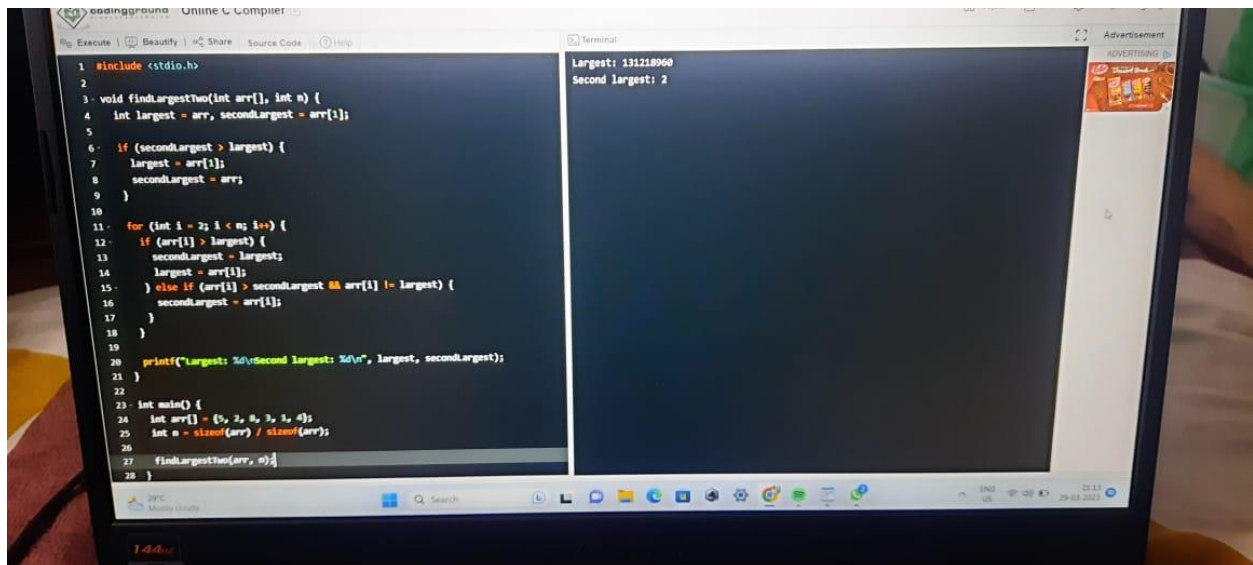
THE SECOND LARGEST = 7

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



The screenshot shows a web-based C compiler interface. The left pane displays the source code, and the right pane shows the terminal output.

```
1 #include <stdio.h>
2
3 void findLargestTwo(int arr[], int n) {
4     int largest = arr[0], secondLargest = arr[1];
5
6     if (secondLargest > largest) {
7         largest = arr[1];
8         secondLargest = arr[0];
9     }
10
11     for (int i = 2; i < n; i++) {
12         if (arr[i] > largest) {
13             secondLargest = largest;
14             largest = arr[i];
15         } else if (arr[i] > secondLargest && arr[i] != largest) {
16             secondLargest = arr[i];
17         }
18     }
19
20     printf("Largest: %d\nSecond largest: %d\n", largest, secondLargest);
21 }
22
23 int main() {
24     int arr[] = {2, 2, 8, 5, 7, 4};
25     int n = sizeof(arr) / sizeof(arr[0]);
26
27     findLargestTwo(arr, n);
28 }
```

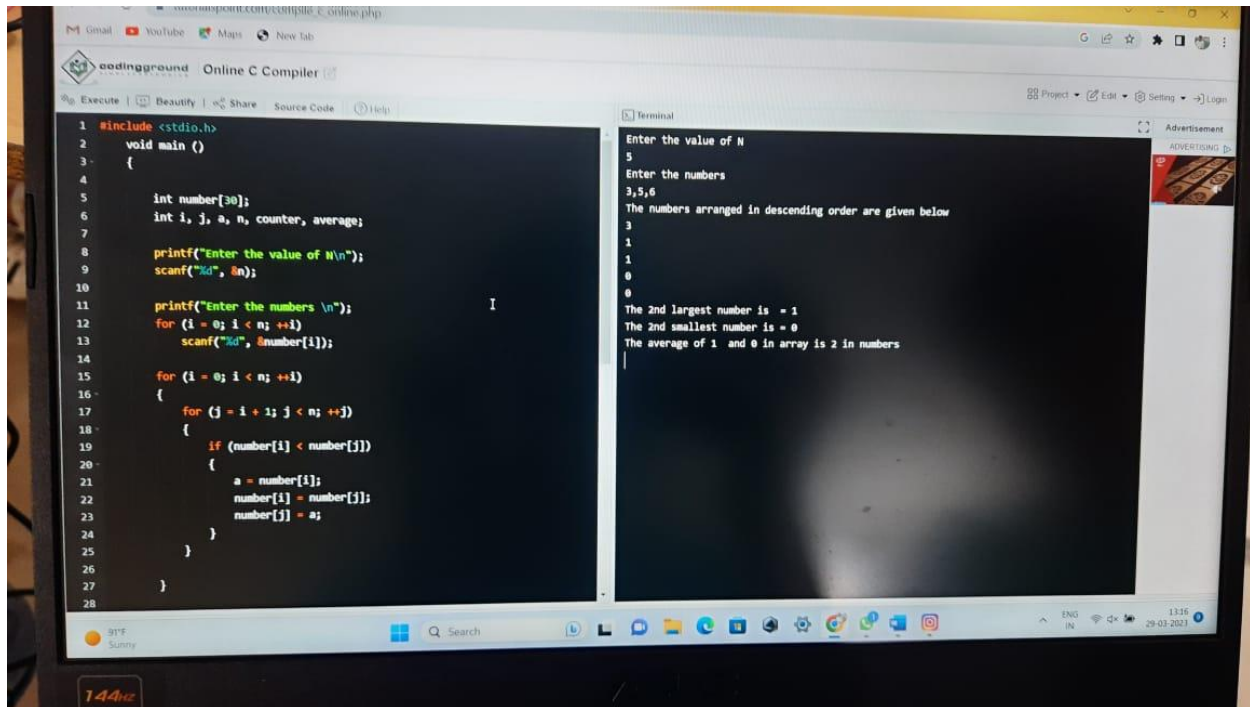
The terminal output on the right shows:

```
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.



The screenshot shows an online C compiler interface. The left pane displays the source code, and the right pane shows the terminal output.

```
1 #include <stdio.h>
2 void main ()
3 {
4     int number[30];
5     int i, j, a, n, counter, average;
6
7     printf("Enter the value of N\n");
8     scanf("%d", &n);
9
10    printf("Enter the numbers \n");
11    for (i = 0; i < n; ++i)
12        scanf("%d", &number[i]);
13
14    for (i = 0; i < n; ++i)
15    {
16        for (j = i + 1; j < n; ++j)
17        {
18            if (number[i] < number[j])
19            {
20                a = number[i];
21                number[i] = number[j];
22                number[j] = a;
23            }
24        }
25    }
26
27 }
28
```

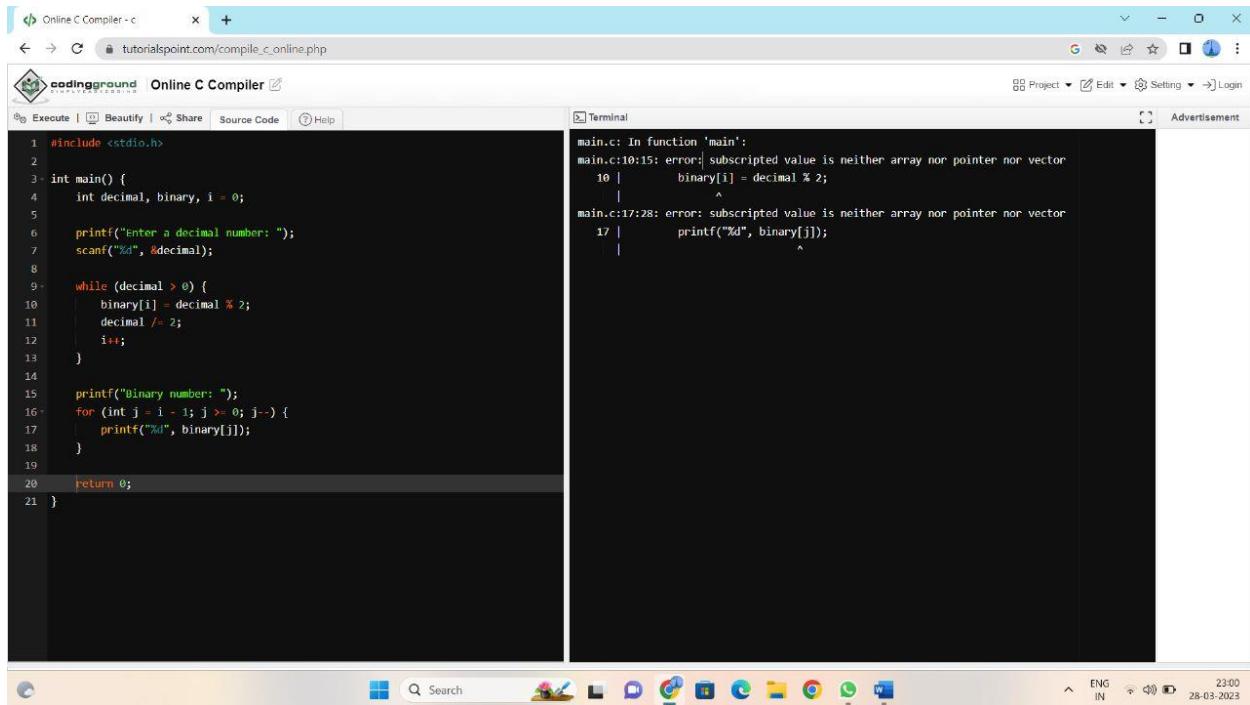
The terminal output shows the following sequence of interactions:

```
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 $200-10=190$



The screenshot shows a web browser window with the URL `tutorialspoint.com/compile_c_online.php`. The page title is "Online C Compiler". The interface includes a code editor on the left and a terminal on the right. The code in the editor is as follows:

```
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 }
```

The terminal on the right shows the following error messages:

```
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]);
   |                   ^
```

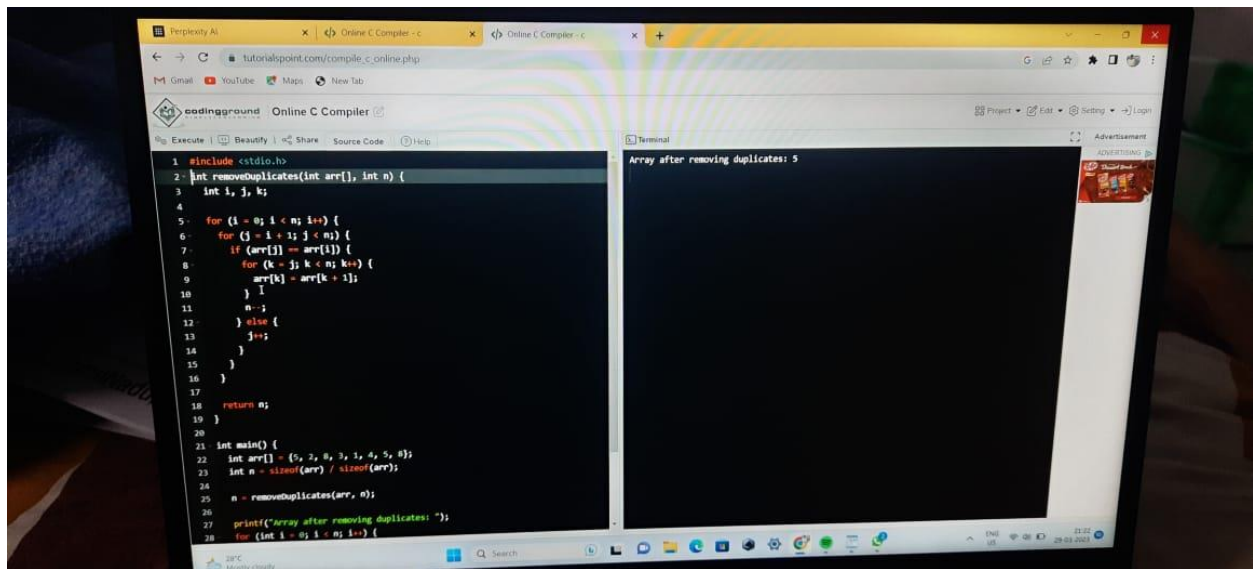
5 C program to remove duplicate elements in an Array?
An array is a collection of similar data elements stored in a contiguous memory location.

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

Example:

Input Array: 1,2,4,5,4,2,7,5

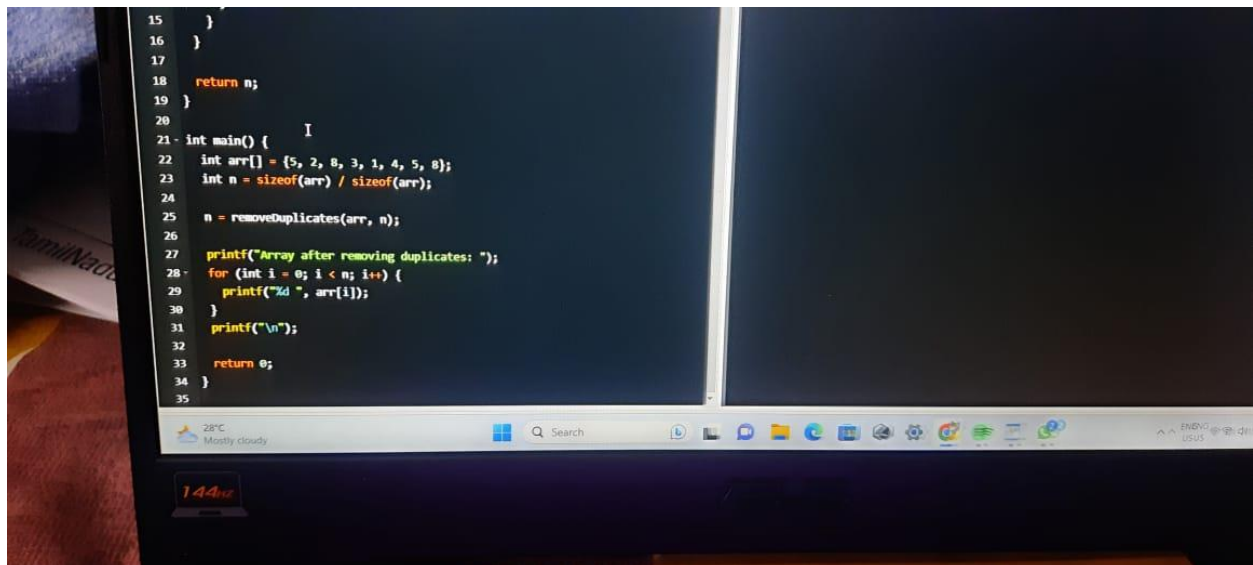
Output: Resultant Array after removing duplicates: 1,2,4,5,7



The screenshot shows a web browser window with an online C compiler. The code in the editor is as follows:

```
1 #include <stdio.h>
2 int removeDuplicates(int arr[], int n) {
3     int i, j, k;
4
5     for (i = 0; i < n; i++) {
6         for (j = i + 1; j < n; j++) {
7             if (arr[i] == arr[j]) {
8                 for (k = j; k < n; k++) {
9                     arr[k] = arr[k + 1];
10                }
11                n--;
12            } else {
13                j++;
14            }
15        }
16    }
17
18    return n;
19 }
20
21 int main() {
22     int arr[] = {5, 2, 8, 3, 1, 4, 5, 8};
23     int n = sizeof(arr) / sizeof(arr[0]);
24
25     n = removeDuplicates(arr, n);
26
27     printf("Array after removing duplicates: ");
28     for (int i = 0; i < n; i++) {
29         printf("%d ", arr[i]);
30     }
31     printf("\n");
32
33     return 0;
34 }
```

The terminal output on the right shows: "Array after removing duplicates: 5".

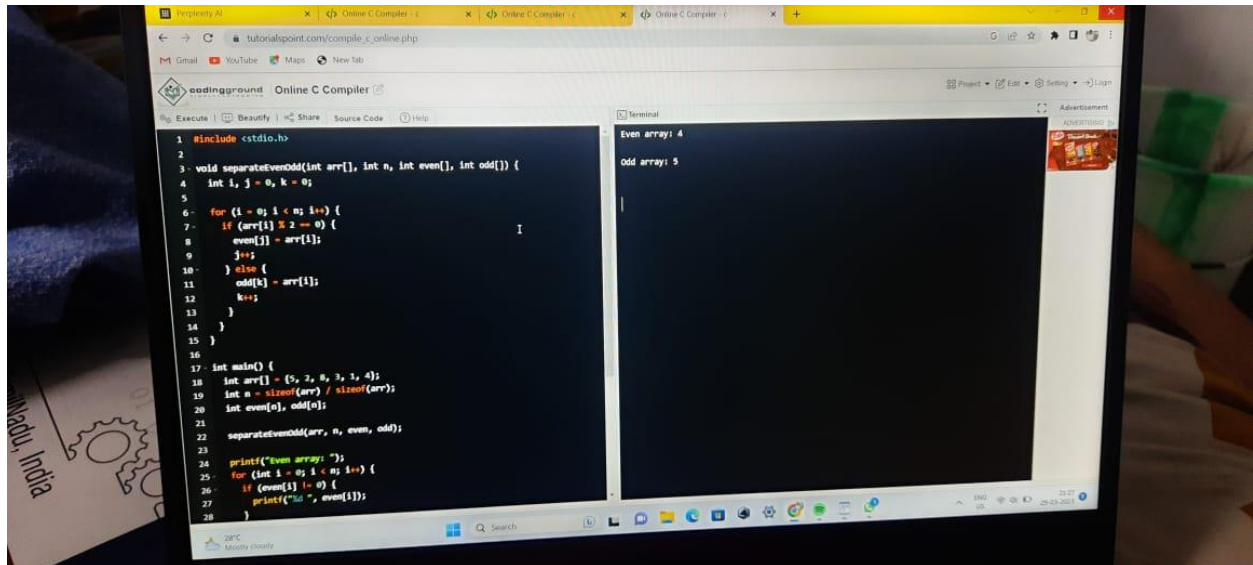


This screenshot shows a different view of the same C program code, focusing on the main function and the array definition:

```
15 }
16 }
17
18 return n;
19 }
20
21 int main() {
22     int arr[] = {5, 2, 8, 3, 1, 4, 5, 8};
23     int n = sizeof(arr) / sizeof(arr[0]);
24
25     n = removeDuplicates(arr, n);
26
27     printf("Array after removing duplicates: ");
28     for (int i = 0; i < n; i++) {
29         printf("%d ", arr[i]);
30     }
31     printf("\n");
32
33     return 0;
34 }
```

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.

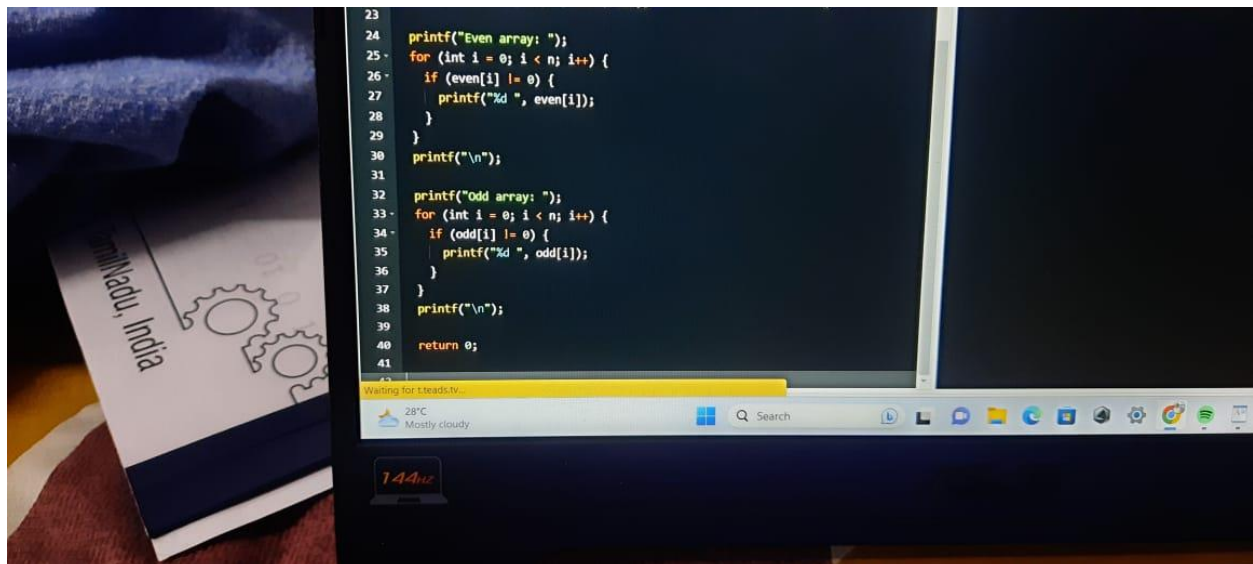


The screenshot shows a web browser with the 'Online C Compiler' interface. The code editor on the left contains the following C program:

```
1 #include <stdio.h>
2
3 void separateEvenOdd(int arr[], int n, int even[], int odd[]) {
4     int i, j = 0, k = 0;
5
6     for (i = 0; i < n; i++) {
7         if (arr[i] % 2 == 0) {
8             even[j] = arr[i];
9             j++;
10        } else {
11            odd[k] = arr[i];
12            k++;
13        }
14    }
15 }
16
17 int main() {
18     int arr[] = {5, 2, 8, 3, 1, 4};
19     int n = sizeof(arr) / sizeof(arr[0]);
20     int even[n], odd[n];
21     separateEvenOdd(arr, n, even, odd);
22
23     printf("Even array: ");
24     for (int i = 0; i < n; i++) {
25         if (even[i] != 0) {
26             printf("%d ", even[i]);
27         }
28     }
29 }
```

The terminal window on the right displays the output of the program:

```
Even array: 4
Odd array: 5
```



This close-up view of the code editor shows the printing logic for the even and odd arrays. The code is as follows:

```
23
24 printf("Even array: ");
25 for (int i = 0; i < n; i++) {
26     if (even[i] != 0) {
27         printf("%d ", even[i]);
28     }
29 }
30 printf("\n");
31
32 printf("Odd array: ");
33 for (int i = 0; i < n; i++) {
34     if (odd[i] != 0) {
35         printf("%d ", odd[i]);
36     }
37 }
38 printf("\n");
39
40 return 0;
41
```

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.

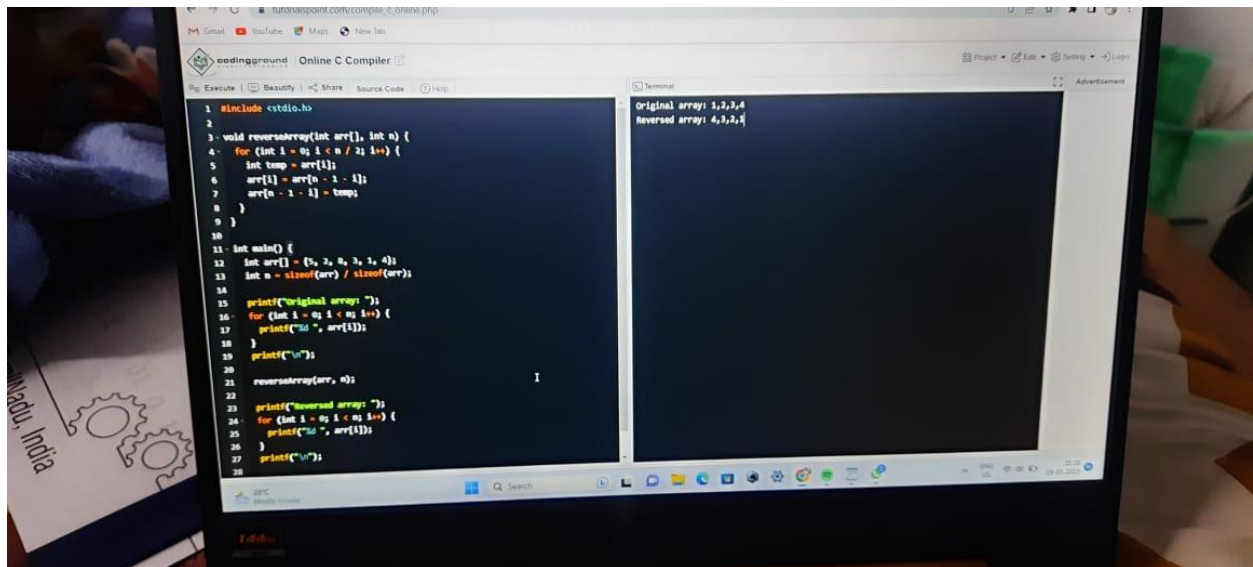
Examples:

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

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

Input array: [3,2,1]

Reversed array: [1,2,3]



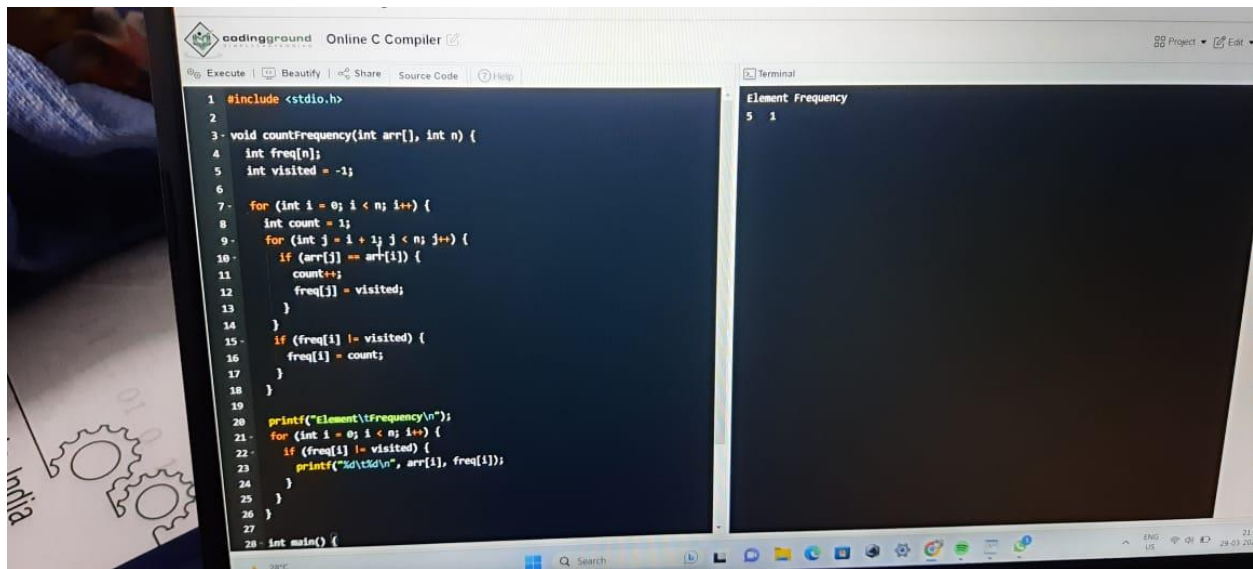
The screenshot shows a web browser window with an online C compiler. The code in the editor is as follows:

```
1 #include <stdio.h>
2
3 void reversearray(int arr[], int n) {
4     for (int i = 0; i < n / 2; i++) {
5         int temp = arr[i];
6         arr[i] = arr[n - 1 - i];
7         arr[n - 1 - i] = temp;
8     }
9 }
10
11 int main() {
12     int arr[] = {1, 2, 3, 4};
13     int n = sizeof(arr) / sizeof(arr[0]);
14
15     printf("Original array: ");
16     for (int i = 0; i < n; i++) {
17         printf("%d ", arr[i]);
18     }
19     printf("\n");
20
21     reversearray(arr, n);
22
23     printf("Reversed array: ");
24     for (int i = 0; i < n; i++) {
25         printf("%d ", arr[i]);
26     }
27     printf("\n");
28 }
```

The terminal output on the right shows:

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

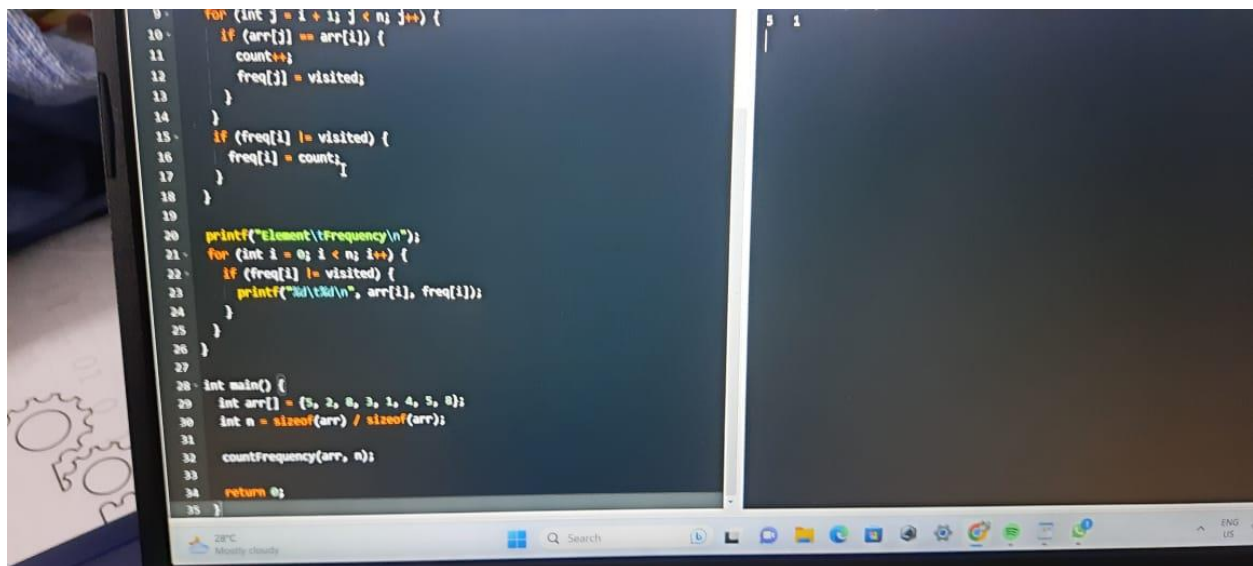
8 Write a program in C to count the frequency of each element of an array.



```
1 #include <stdio.h>
2
3 void countFrequency(int arr[], int n) {
4     int freq[n];
5     int visited = -1;
6
7     for (int i = 0; i < n; i++) {
8         int count = 1;
9         for (int j = i + 1; j < n; j++) {
10             if (arr[j] == arr[i]) {
11                 count++;
12                 freq[j] = visited;
13             }
14         }
15         if (freq[i] != visited) {
16             freq[i] = count;
17         }
18     }
19
20     printf("Element\tFrequency\n");
21     for (int i = 0; i < n; i++) {
22         if (freq[i] != visited) {
23             printf("%d\t%d\n", arr[i], freq[i]);
24         }
25     }
26 }
27
28 int main() {
```

Terminal

```
Element Frequency
5 1
```



```
9         for (int j = i + 1; j < n; j++) {
10             if (arr[j] == arr[i]) {
11                 count++;
12                 freq[j] = visited;
13             }
14         }
15         if (freq[i] != visited) {
16             freq[i] = count;
17         }
18     }
19
20     printf("Element\tFrequency\n");
21     for (int i = 0; i < n; i++) {
22         if (freq[i] != visited) {
23             printf("%d\t%d\n", arr[i], freq[i]);
24         }
25     }
26 }
27
28 int main() {
29     int arr[] = {5, 2, 0, 3, 1, 4, 5, 0};
30     int n = sizeof(arr) / sizeof(arr[0]);
31
32     countFrequency(arr, n);
33
34     return 0;
35 }
```

Terminal

```
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

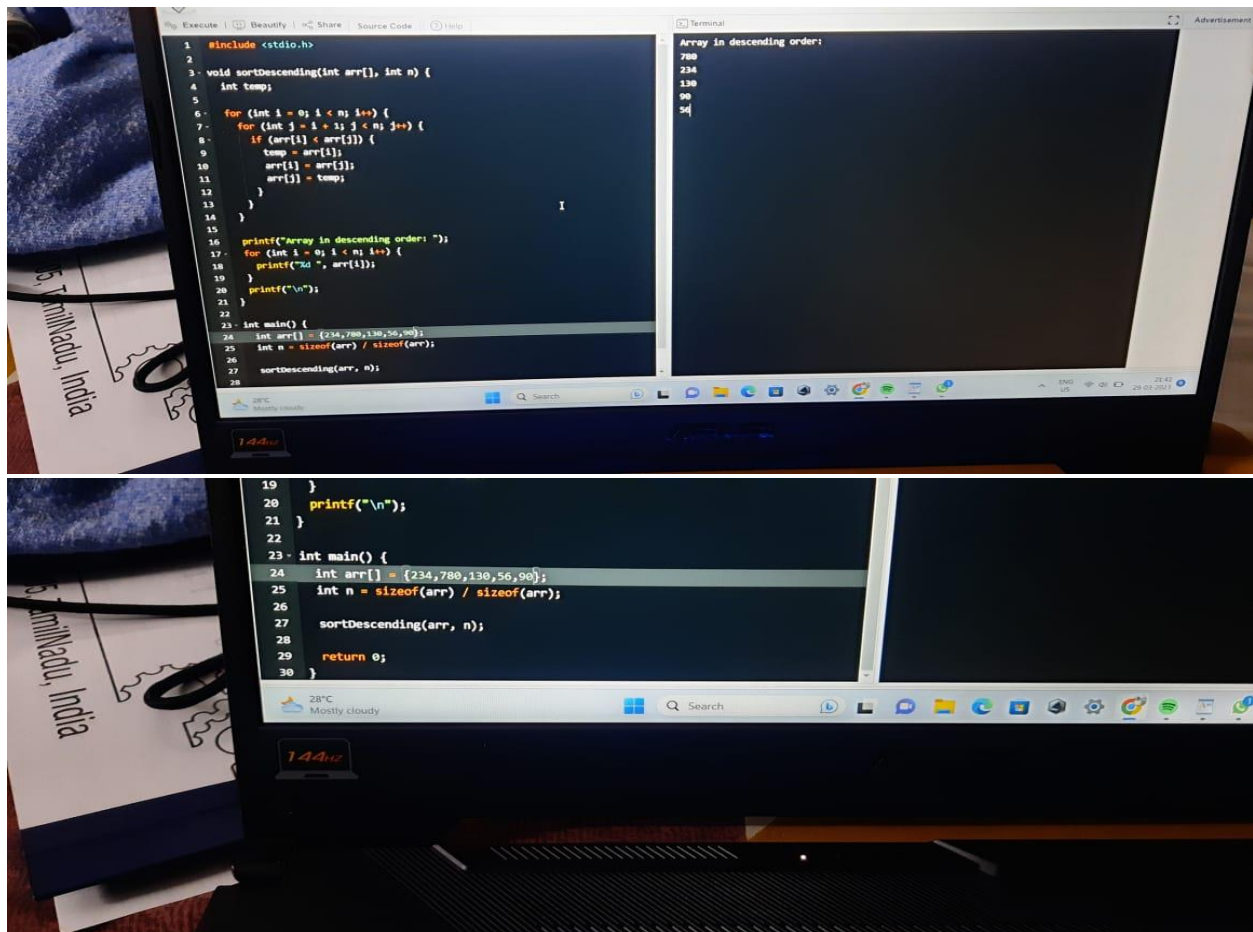
780

234

130

90

56



The image consists of two screenshots of a laptop screen displaying a C++ program. The program sorts an array in descending order using a bubble sort algorithm. The first screenshot shows the code and the output of the program. The second screenshot shows a different part of the code, specifically the main function and the array definition.

```
1 #include <stdio.h>
2
3 void sortDescending(int arr[], int n) {
4     int temp;
5
6     for (int i = 0; i < n; i++) {
7         for (int j = i + 1; j < n; j++) {
8             if (arr[i] < arr[j]) {
9                 temp = arr[i];
10                arr[i] = arr[j];
11                arr[j] = temp;
12            }
13        }
14    }
15
16    printf("Array in descending order: ");
17    for (int i = 0; i < n; i++) {
18        printf("%d ", arr[i]);
19    }
20    printf("\n");
21 }
22
23 int main() {
24     int arr[] = {234, 780, 130, 56, 90};
25     int n = sizeof(arr) / sizeof(arr[0]);
26     sortDescending(arr, n);
27 }
28
```

Array in descending order:
780
234
130
90
56

```
19 }
20 printf("\n");
21 }
22
23 int main() {
24     int arr[] = {234, 780, 130, 56, 90};
25     int n = sizeof(arr) / sizeof(arr[0]);
26     sortDescending(arr, n);
27     return 0;
28 }
```

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.

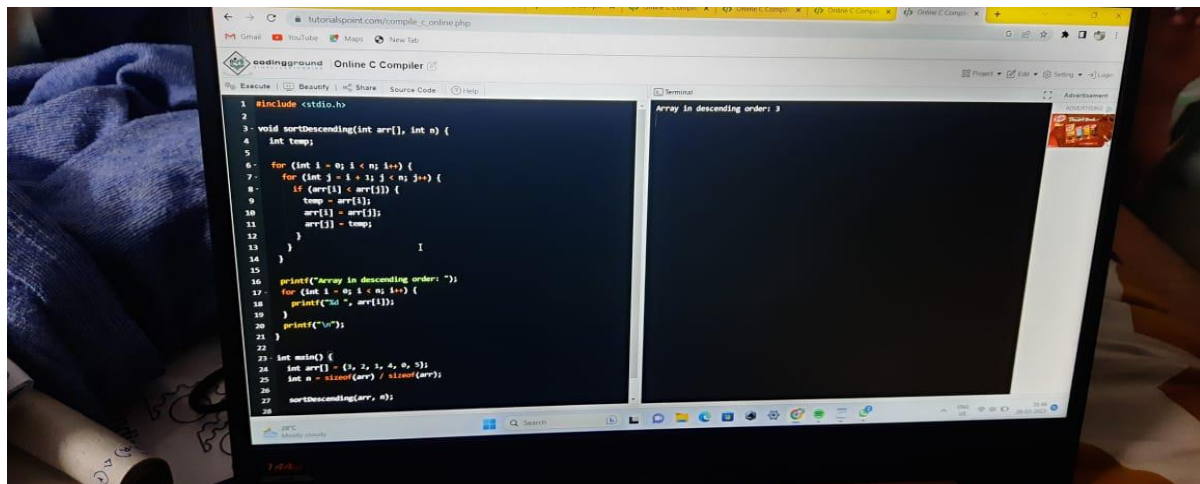
Now there are three options 5, 8 or 9. I

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

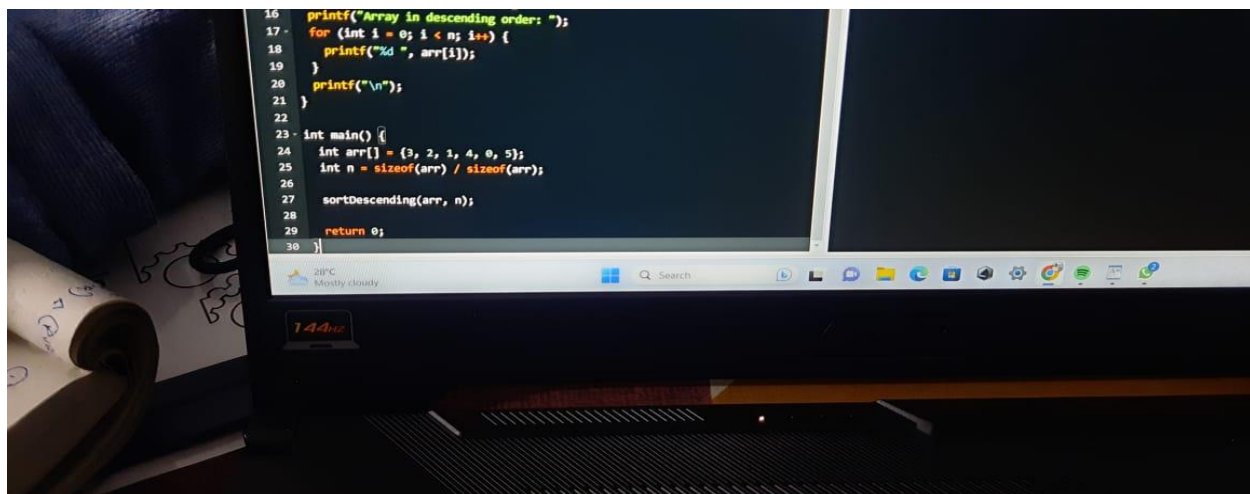
Explanation: In every step a jump is needed so the count of jumps is 10.



The screenshot shows a web browser window with an online C compiler. The code defines a function `sortDescending` that sorts an array in descending order using a bubble sort algorithm. In the `main` function, an array `arr` is initialized with the values {1, 2, 1, 4, 0, 5}, and the function `sortDescending` is called with `arr` and its size `n`. The output of the program is displayed in a terminal window, showing the array in descending order: 5, 4, 2, 1, 0, 1.

```
1 #include <stdio.h>
2
3 void sortDescending(int arr[], int n) {
4     int temp;
5
6     for (int i = 0; i < n; i++) {
7         for (int j = i + 1; j < n; j++) {
8             if (arr[i] < arr[j]) {
9                 temp = arr[i];
10                arr[i] = arr[j];
11                arr[j] = temp;
12            }
13        }
14    }
15
16    printf("Array in descending order: ");
17    for (int i = 0; i < n; i++) {
18        printf("%d ", arr[i]);
19    }
20    printf("\n");
21 }
22
23 int main() {
24     int arr[] = {1, 2, 1, 4, 0, 5};
25     int n = sizeof(arr) / sizeof(arr[0]);
26
27     sortDescending(arr, n);
28 }
29
```

Array in descending order: 5



The screenshot shows a close-up of the code editor in the online C compiler. It displays the `main` function and the `sortDescending` function call. The array `arr` is initialized with {1, 2, 1, 4, 0, 5} and the function is called with `arr` and `n`. The output of the program is displayed in a terminal window, showing the array in descending order: 5, 4, 2, 1, 0, 1.

```
16 printf("Array in descending order: ");
17 for (int i = 0; i < n; i++) {
18     printf("%d ", arr[i]);
19 }
20 printf("\n");
21 }
22
23 int main() {
24     int arr[] = {1, 2, 1, 4, 0, 5};
25     int n = sizeof(arr) / sizeof(arr[0]);
26
27     sortDescending(arr, n);
28 }
29 return 0;
30 }
```