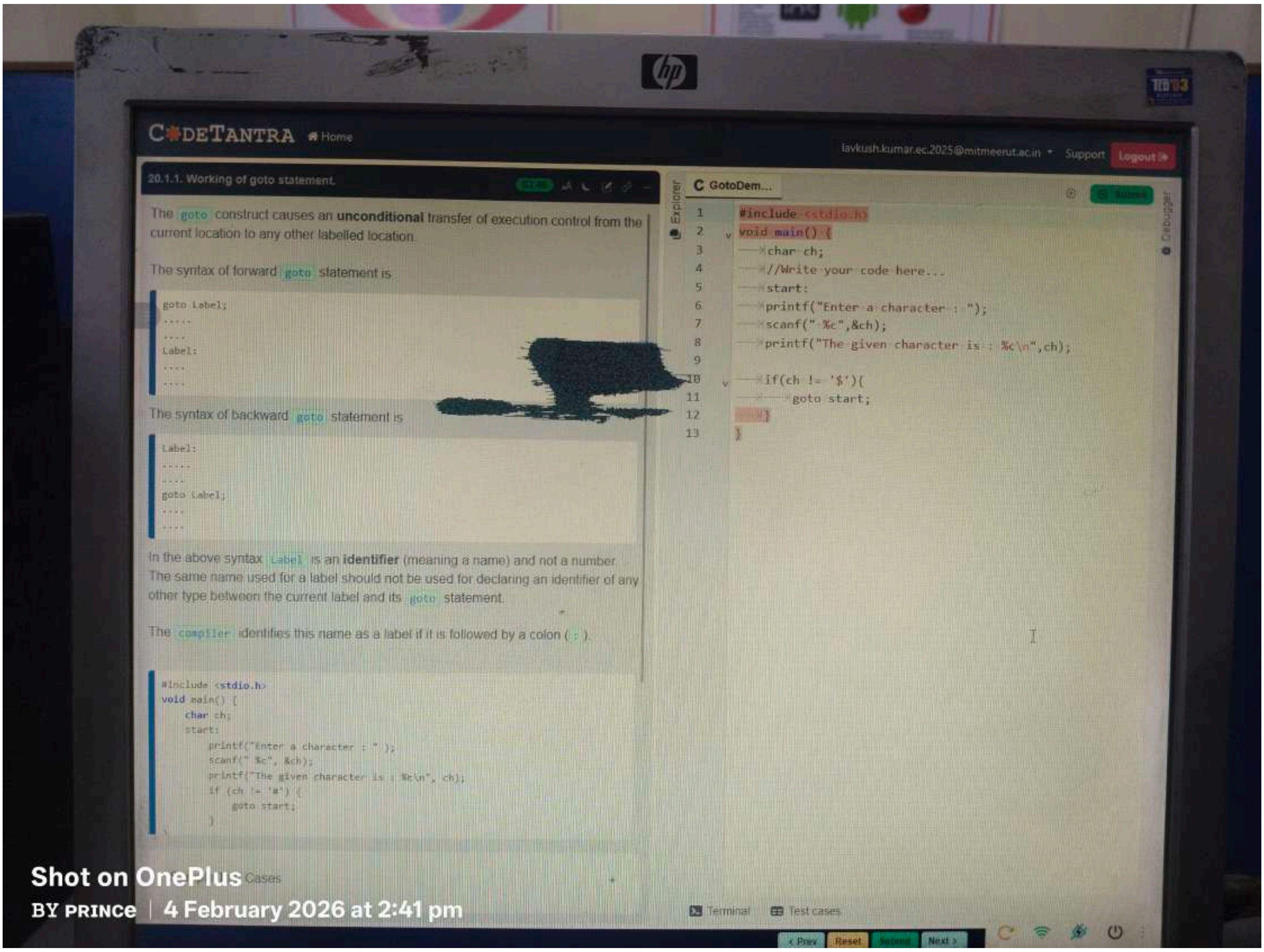


20.1.1. Working of goto statement

The goto construct causes an **unconditional** transfer of execution control from the current location to any other labelled location.

The syntax of forward goto statement is

```
goto Label;
.....
....
Label:
....
....
```

[redacted]

The syntax of backward goto statement is

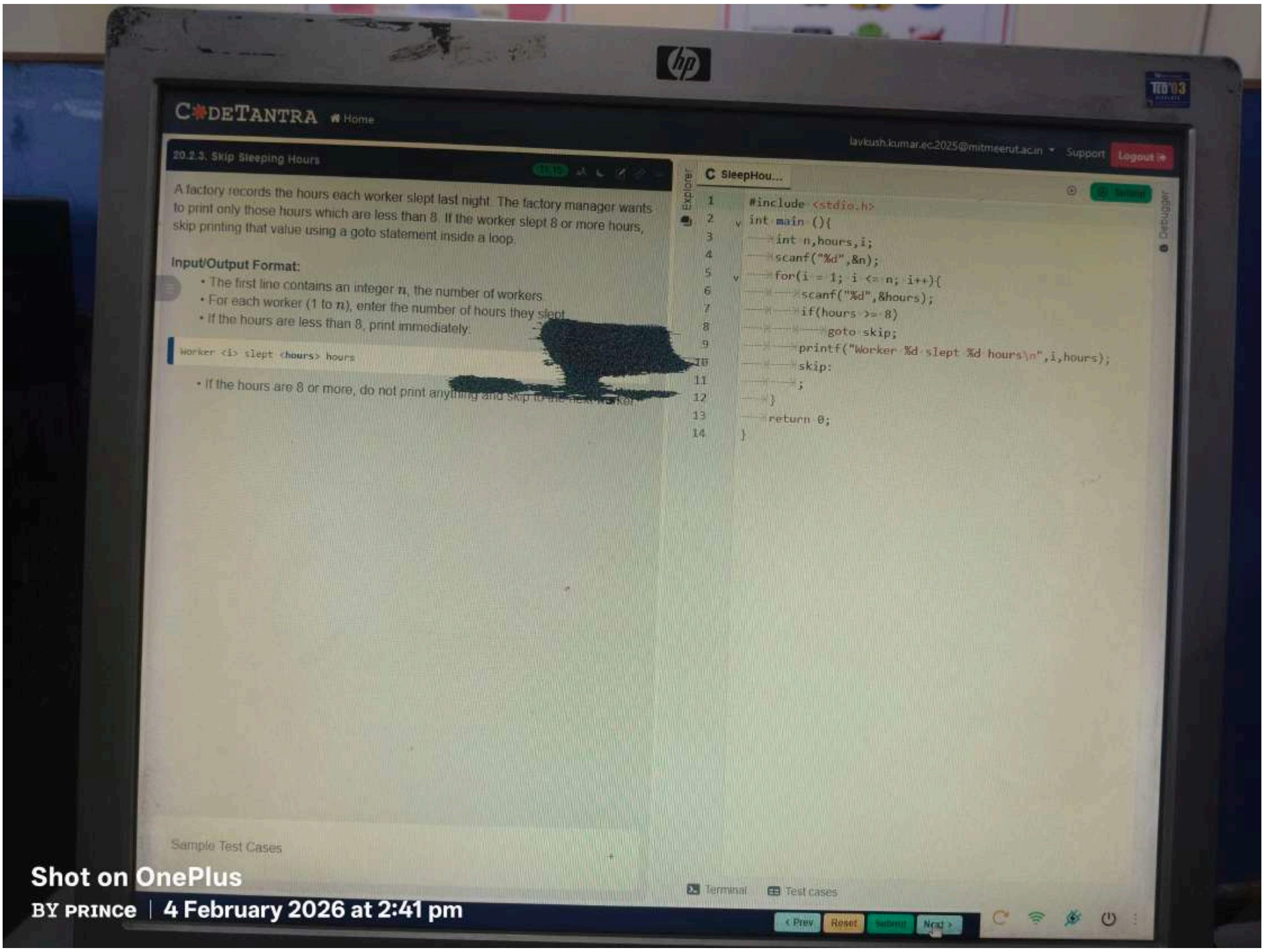
```
Label:
.....
....
goto Label;
....
....
```

In the above syntax Label is an **identifier** (meaning a name) and not a number. The same name used for a label should not be used for declaring an identifier of any other type between the current label and its goto statement.

The compiler identifies this name as a label if it is followed by a colon (:).

```
#include <stdio.h>
void main() {
    char ch;
    start:
        printf("Enter a character : " );
        scanf(" %c", &ch);
        printf("The given character is : %c\n", ch);
        if (ch != '#') {
            goto start;
        }
```

```
#include <stdio.h>
void main() {
	char ch;
	//Write your code here...
	start:
	printf("Enter a character : ");
	scanf(" %c",&ch);
	printf("The given character is : %c\n",ch);

	if(ch != '$'){
		goto start;
	}
}
```



20.2.3. Skip Sleeping Hours

A factory records the hours each worker slept last night. The factory manager wants to print only those hours which are less than 8. If the worker slept 8 or more hours, skip printing that value using a goto statement inside a loop.

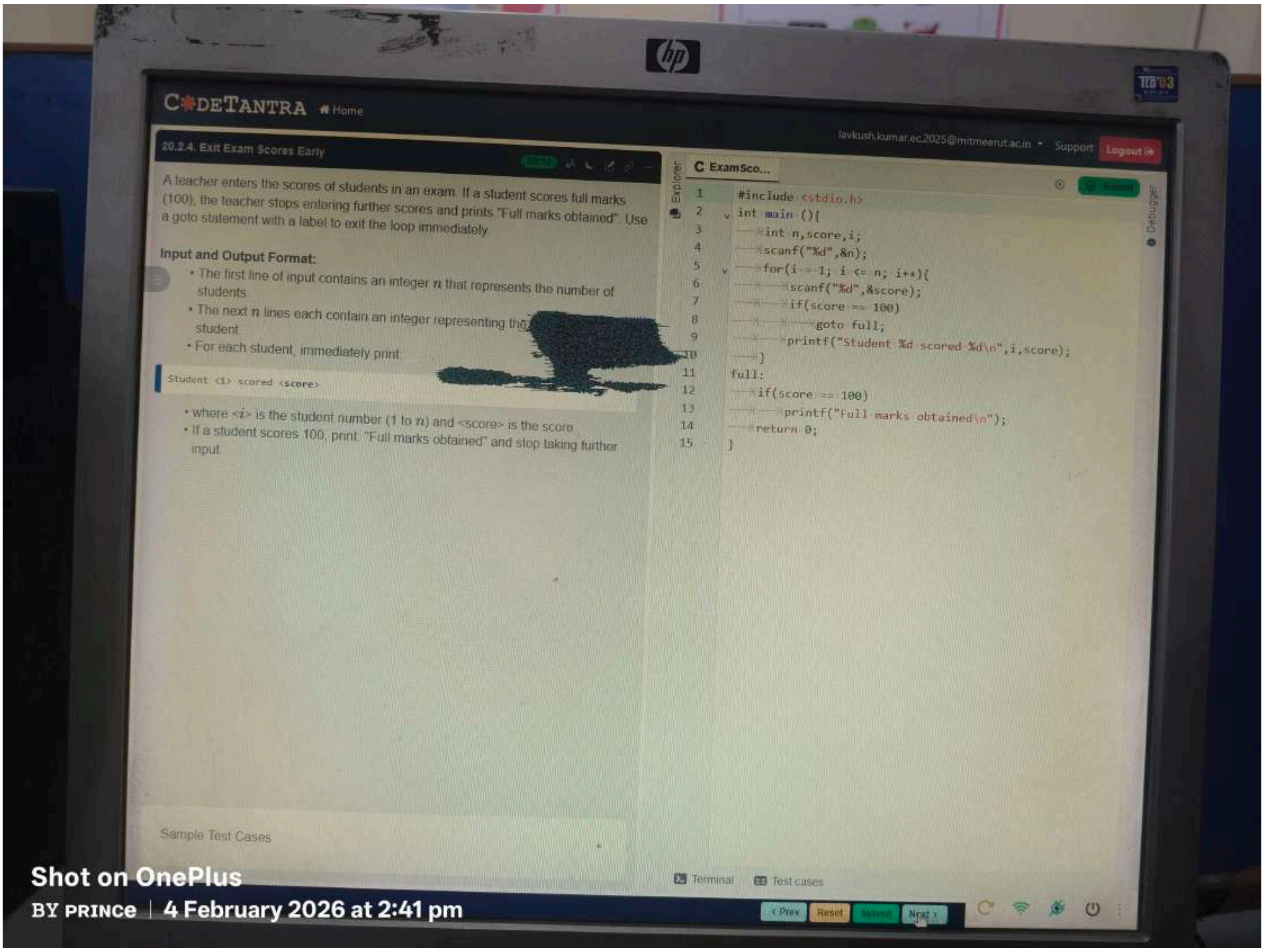
Input/Output Format:

- The first line contains an integer n, the number of workers.
- For each worker (1 to n), enter the number of hours they slept.
- If the hours are less than 8, print immediately.

```
Worker <i> slept <hours> hours
```

- If the hours are 8 or more, do not print any[redacted]

```c
#include <stdio.h>
int main (){
	int n,hours,i;
	scanf("%d",&n);
	for(i = 1; i <= n; i++){
		scanf("%d",&hours);
		if(hours >= 8)
			goto skip;
		printf("Worker %d slept %d hours\n",i,hours);
		skip:
		;
	}
	return 0;
}
```



20.2.4. Exit Exam Scores Early

A teacher enters the scores of students in an exam. If a student scores full marks (100), the teacher stops entering further scores and prints "Full marks obtained". Use a goto statement with a label to exit the loop immediately.

Input and Output Format:

- The first line of input contains an integer n that represents the number of students.
- The next n lines each contain an integer representing the [redacted] student.
- For each student, immediately print:

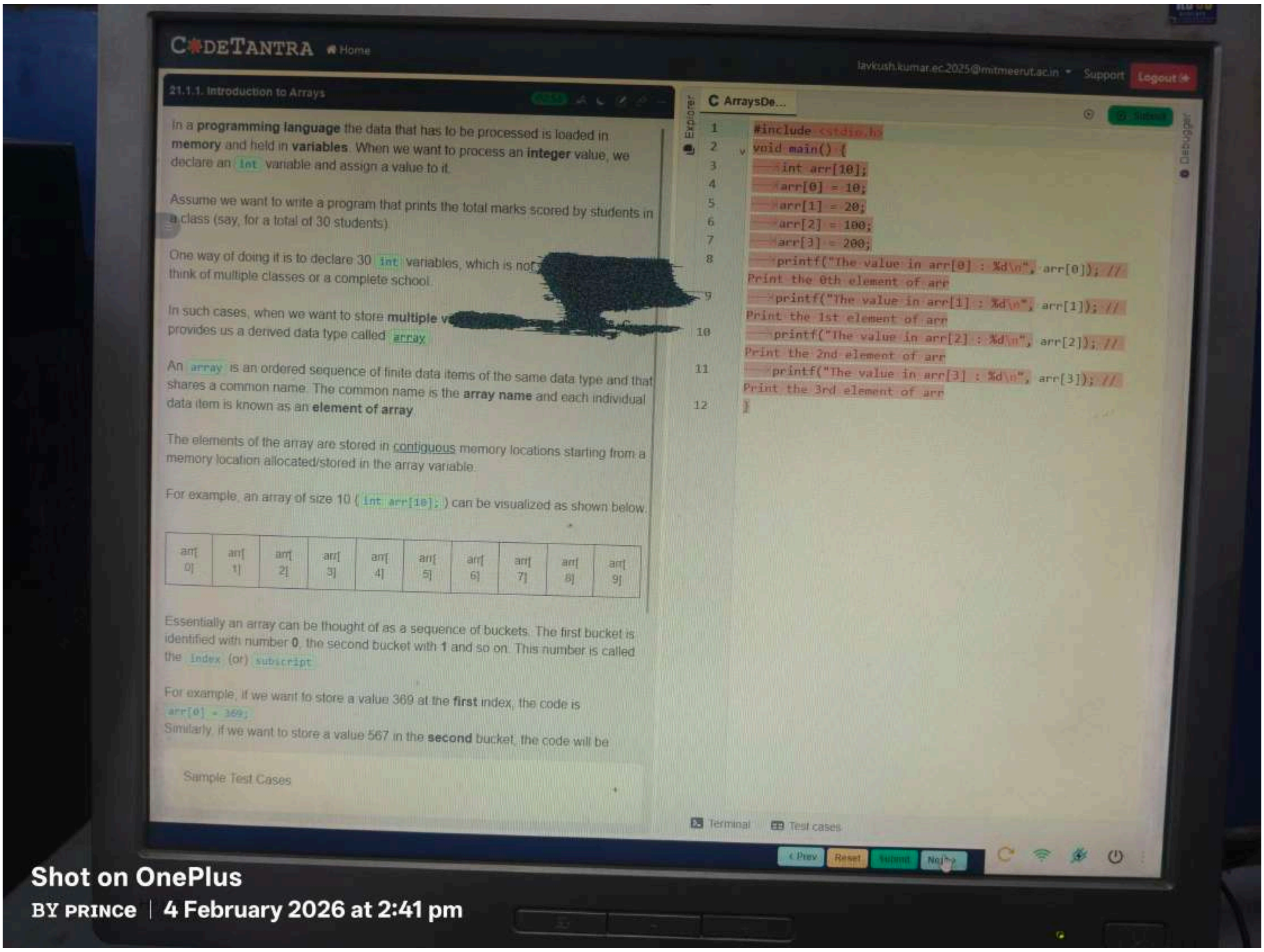
```
Student <i> scored <score>
```

- where $<i>$ is the student number (1 to n) and <score> is the score.
- If a student scores 100, print "Full marks obtained" and stop taking further input.

Sample Test Cases

```c
#include <stdio.h>
int main (){
	int n,score,i;
	scanf("%d",&n);
	for(i = 1; i <= n; i++){
		scanf("%d",&score);
		if(score == 100)
			goto full;
		printf("Student %d scored %d\n",i,score);
	}
full:
	if(score == 100)
		printf("Full marks obtained\n");
	return 0;
}
```

Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:41 pm



21.1.1. Introduction to Arrays

In a **programming language** the data that has to be processed is loaded in **memory** and held in **variables**. When we want to process an **integer** value, we declare an `int` variable and assign a value to it.

Assume we want to write a program that prints the total marks scored by students in a class (say, for a total of 30 students).

One way of doing it is to declare 30 `int` variables, which is not [redacted] think of multiple classes or a complete school.

In such cases, when we want to store **multiple v** provides us a derived data type called `array`.

An `array` is an ordered sequence of finite data items of the same data type and that shares a common name. The common name is the **array name** and each individual data item is known as an **element of array**.

The elements of the array are stored in contiguous memory locations starting from a memory location allocated/stored in the array variable.

For example, an array of size 10 (`int arr[10];`) can be visualized as shown below.

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]	arr[7]	arr[8]	arr[9]

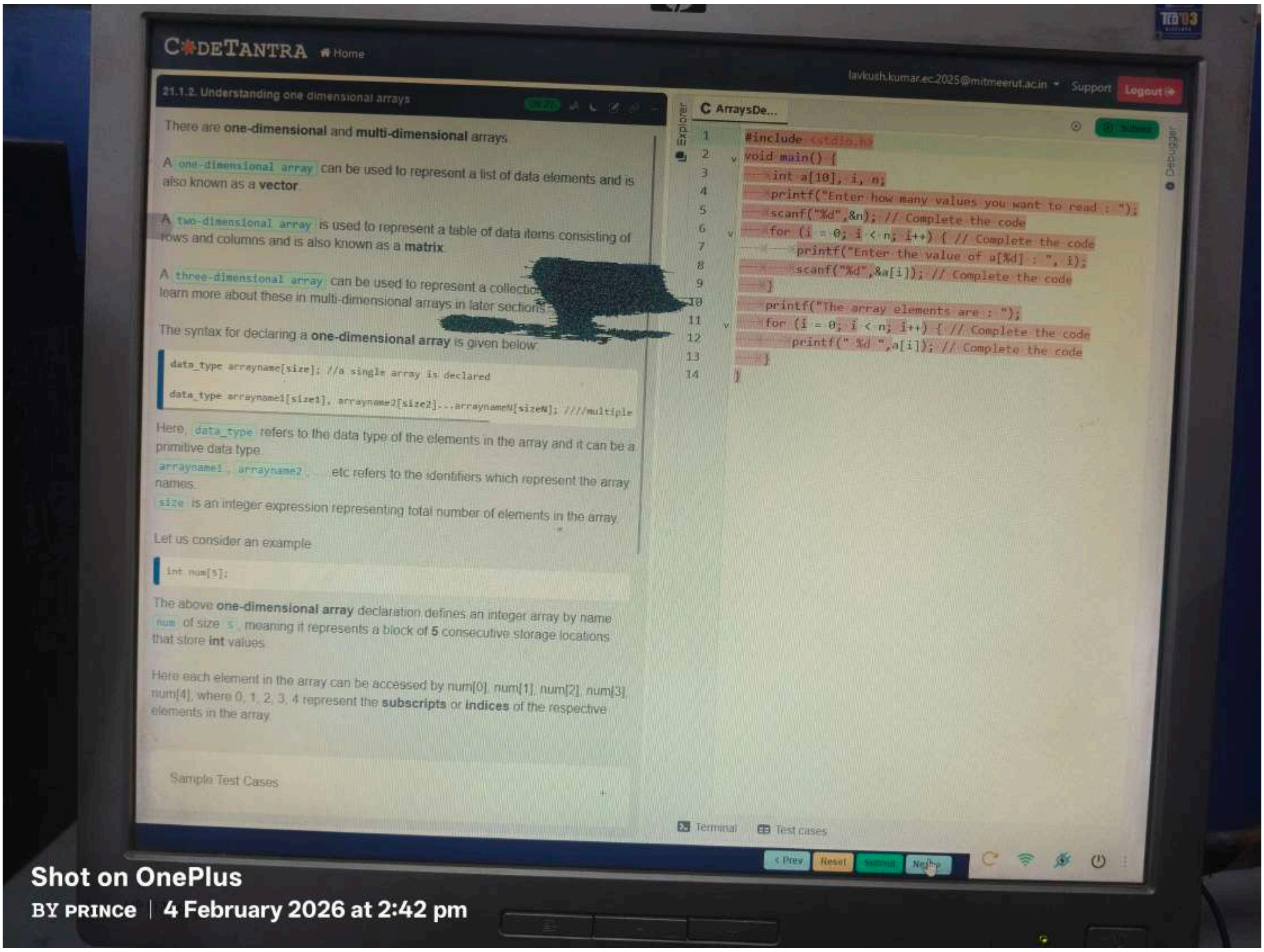
Essentially an array can be thought of as a sequence of buckets. The first bucket is identified with number **0**, the second bucket with **1** and so on. This number is called the `index` (or) `subscript`.

For example, if we want to store a value 369 at the **first** index, the code is `arr[0] = 369;`
Similarly, if we want to store a value 567 in the **second** bucket, the code will be

Sample Test Cases

C ArraysDe...

```
#include <stdio.h>
void main() {
	int arr[10];
	arr[0] = 10;
	arr[1] = 20;
	arr[2] = 100;
	arr[3] = 200;
	printf("The value in arr[0] : %d\n", arr[0]); // Print the 0th element of arr
	printf("The value in arr[1] : %d\n", arr[1]); // Print the 1st element of arr
	printf("The value in arr[2] : %d\n", arr[2]); // Print the 2nd element of arr
	printf("The value in arr[3] : %d\n", arr[3]); // Print the 3rd element of arr
}
```



21.1.2. Understanding one dimensional arrays

There are **one-dimensional** and **multi-dimensional** arrays.

A `one-dimensional array` can be used to represent a list of data elements and is also known as a **vector**.

A `two-dimensional array` is used to represent a table of data items consisting of rows and columns and is also known as a **matrix**.

A `three-dimensional array` can be used to represent a collectio [redacted] learn more about these in multi-dimensional arrays in later sections

The syntax for declaring a **one-dimensional array** is given below:

```
data_type arrayname[size]; //a single array is declared
data_type arrayname1[size1], arrayname2[size2]...arraynameN[sizeN]; ////multiple
```

Here, `data_type` refers to the data type of the elements in the array and it can be a primitive data type.

`arrayname1`, `arrayname2`, ... etc refers to the identifiers which represent the array names.

`size` is an integer expression representing total number of elements in the array.

Let us consider an example

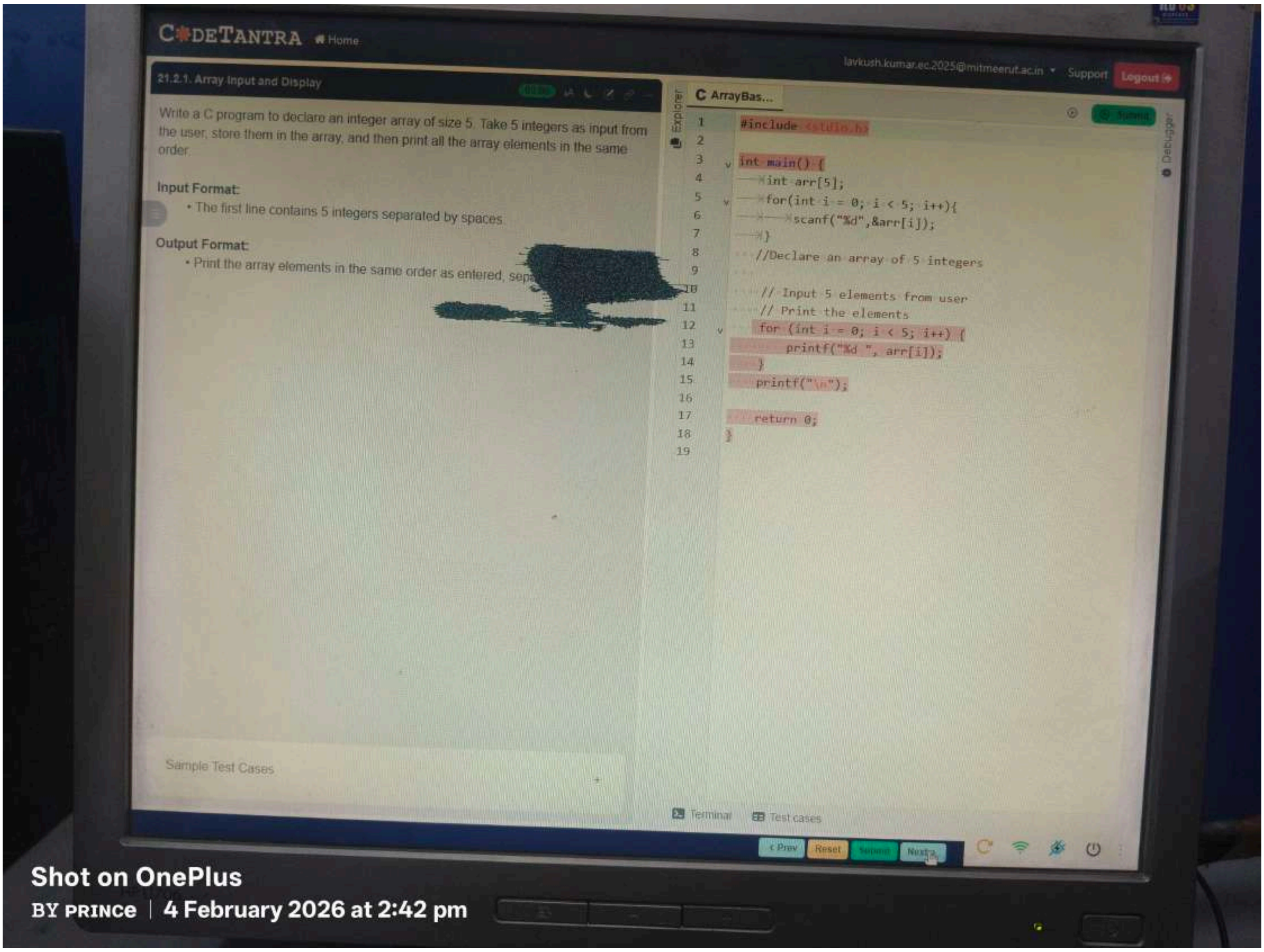
```
int num[5];
```

The above **one-dimensional array** declaration defines an integer array by name `num` of size `5`, meaning it represents a block of **5** consecutive storage locations that store **int** values.

Here each element in the array can be accessed by num[0], num[1], num[2], num[3], num[4], where 0, 1, 2, 3, 4 represent the **subscripts** or **indices** of the respective elements in the array.

Sample Test Cases

```c
#include <stdio.h>
void main() {
	int a[10], i, n;
	printf("Enter how many values you want to read : ");
	scanf("%d",&n); // Complete the code
	for (i = 0; i < n; i++) { // Complete the code
		printf("Enter the value of a[%d] : ", i);
		scanf("%d",&a[i]); // Complete the code
	}
	printf("The array elements are : ");
	for (i = 0; i < n; i++) { // Complete the code
		printf(" %d ",a[i]); // Complete the code
	}
}
```



21.2.1. Array Input and Display

Write a C program to declare an integer array of size 5. Take 5 integers as input from the user, store them in the array, and then print all the array elements in the same order.

Input Format:

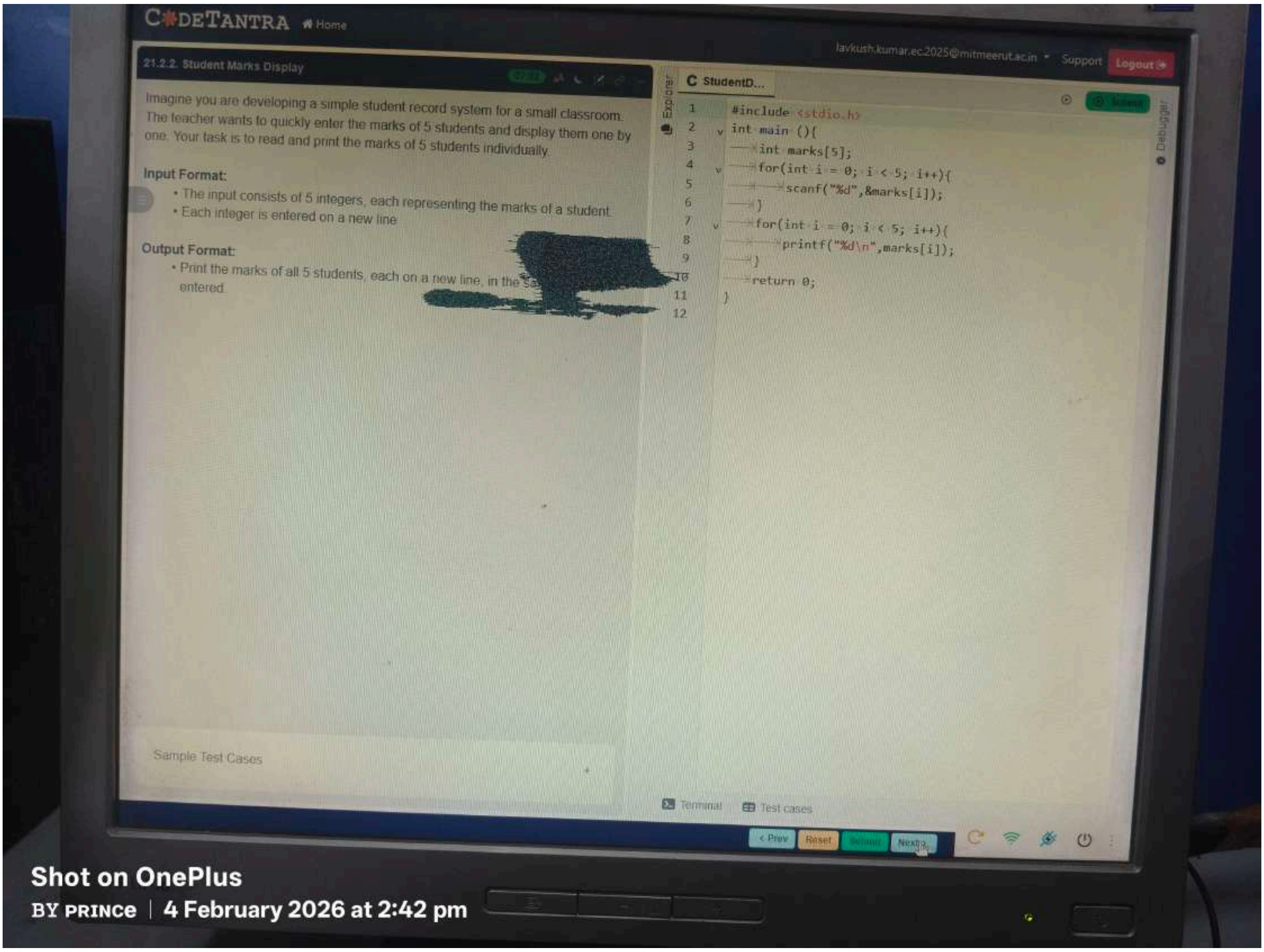
- The first line contains 5 integers separated by spaces

Output Format:

- Print the array elements in the same order as entered, sep[redacted]

```
#include <stdio.h>

int main() {
    int arr[5];
    for(int i = 0; i < 5; i++){
        scanf("%d",&arr[i]);
    }
    //Declare an array of 5 integers

    // Input 5 elements from user
    // Print the elements
    for (int i = 0; i < 5; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:42 pm



21.2.2. Student Marks Display

Imagine you are developing a simple student record system for a small classroom. The teacher wants to quickly enter the marks of 5 students and display them one by one. Your task is to read and print the marks of 5 students individually.

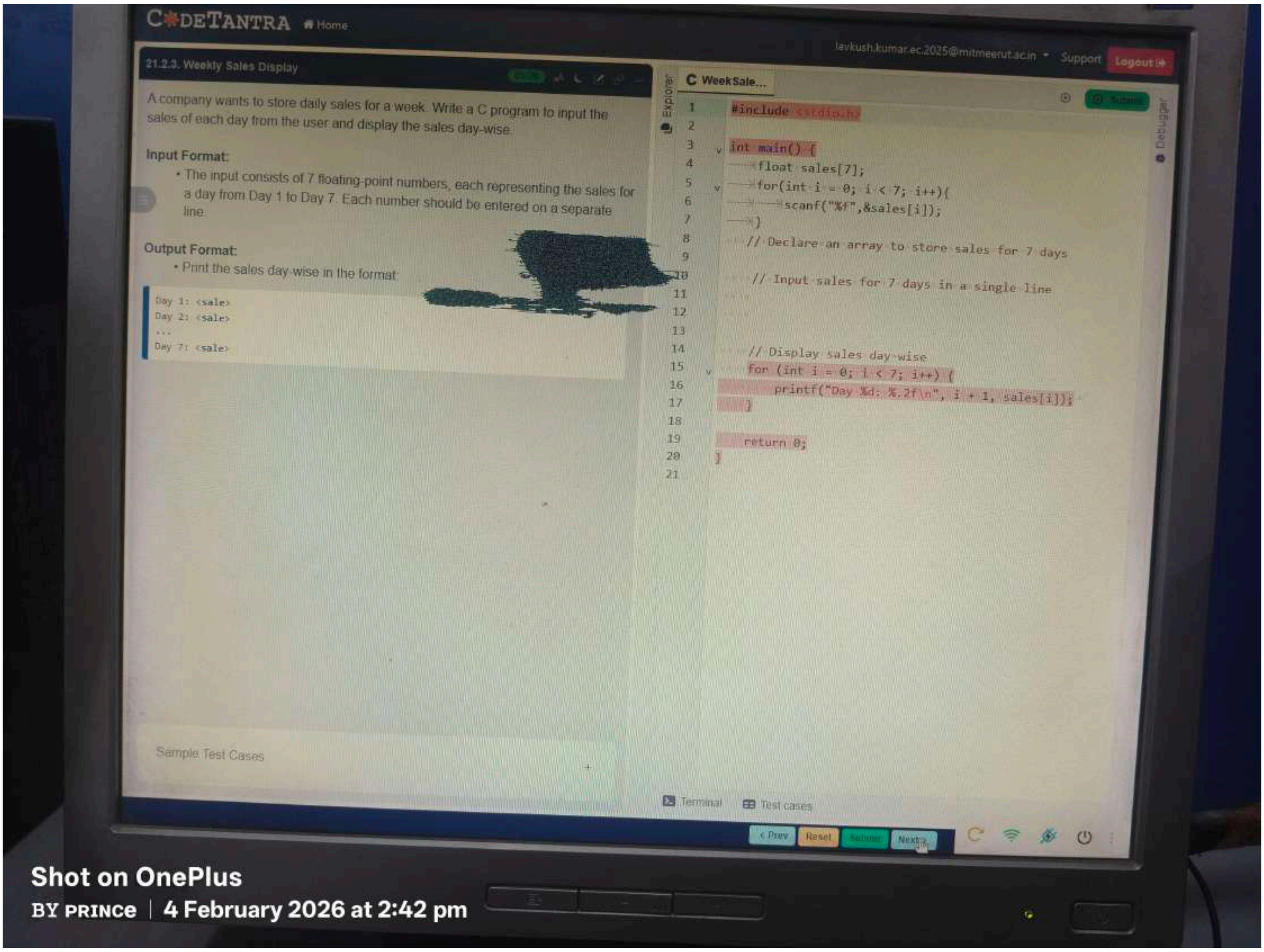
Input Format:

- The input consists of 5 integers, each representing the marks of a student.
- Each integer is entered on a new line.

Output Format:

- Print the marks of all 5 students, each on a new line, in the sa [redacted] entered.

```c
#include <stdio.h>
int main (){
	int marks[5];
	for(int i = 0; i < 5; i++){
		scanf("%d",&marks[i]);
	}
	for(int i = 0; i < 5; i++){
		printf("%d\n",marks[i]);
	}
	return 0;
}
```



21.2.3. Weekly Sales Display

A company wants to store daily sales for a week. Write a C program to input the sales of each day from the user and display the sales day-wise.

Input Format:

- The input consists of 7 floating-point numbers, each representing the sales for a day from Day 1 to Day 7. Each number should be entered on a separate line.

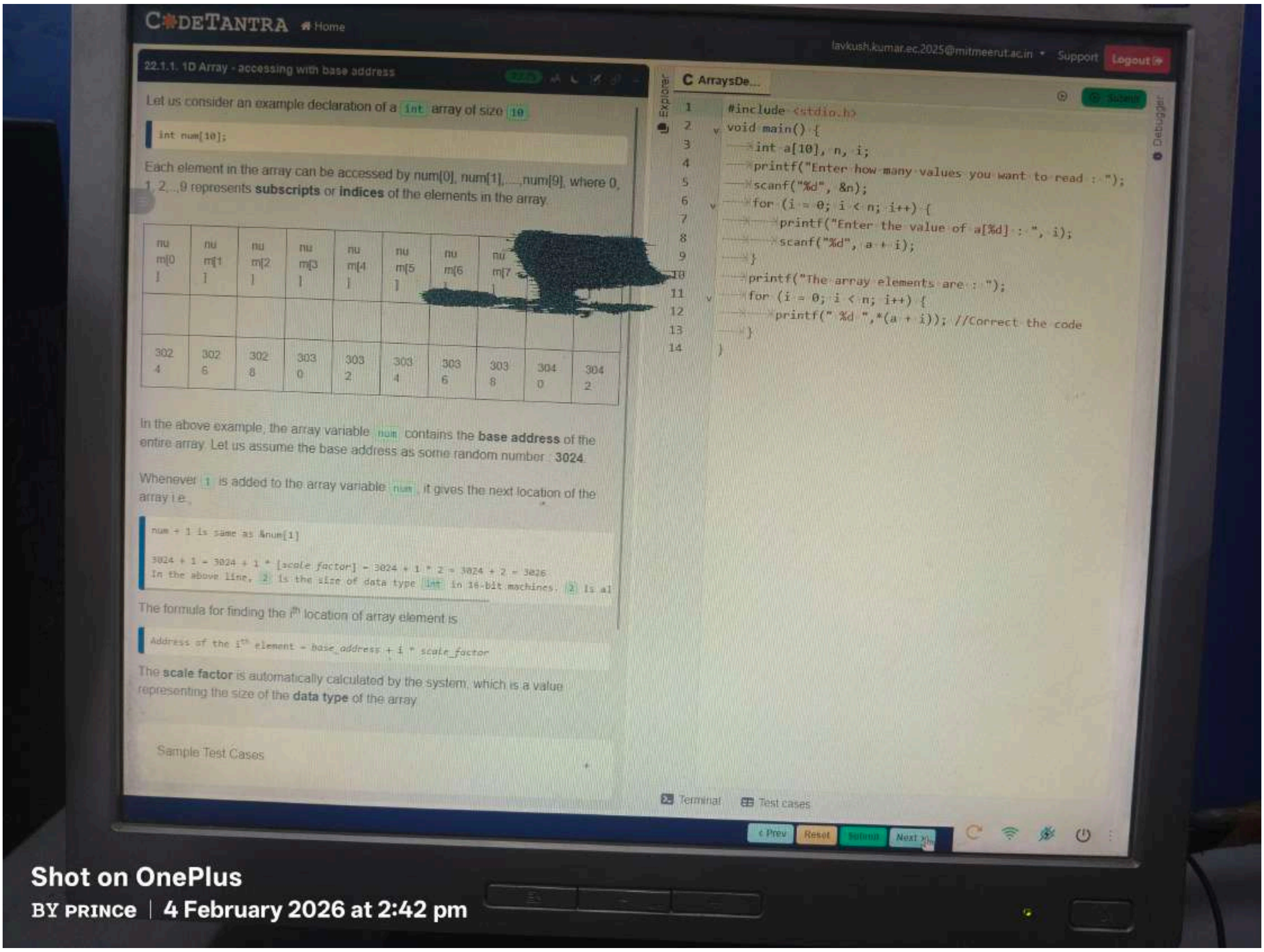
Output Format:

- Print the sales day-wise in the format: [redacted]

```
Day 1: <sale>
Day 2: <sale>
...
Day 7: <sale>
```

C WeekSale...

```c
#include <stdio.h>

int main() {
	float sales[7];
	for(int i = 0; i < 7; i++){
		scanf("%f",&sales[i]);
	}
    // Declare an array to store sales for 7 days

    // Input sales for 7 days in a single line



    // Display sales day-wise
    for (int i = 0; i < 7; i++) {
        printf("Day %d: %.2f\n", i + 1, sales[i]);
    }

    return 0;
}
```



22.1.1. 1D Array - accessing with base address

Let us consider an example declaration of a `int` array of size `10`

```
int num[10];
```

Each element in the array can be accessed by num[0], num[1],...,num[9], where 0, 1, 2,..,9 represents **subscripts** or **indices** of the elements in the array.

num[0]	num[1]	num[2]	num[3]	num[4]	num[5]	num[6]	num[7]	[redacted]	[redacted]
3024	3026	3028	3030	3032	3034	3036	3038	3040	3042

In the above example, the array variable `num` contains the **base address** of the entire array. Let us assume the base address as some random number : **3024**.

Whenever `1` is added to the array variable `num`, it gives the next location of the array i.e.,

```
num + 1 is same as &num[1]

3024 + 1 = 3024 + 1 * [scale factor] = 3024 + 1 * 2 = 3024 + 2 = 3026
In the above line, 2 is the size of data type int in 16-bit machines. 2 is al
```

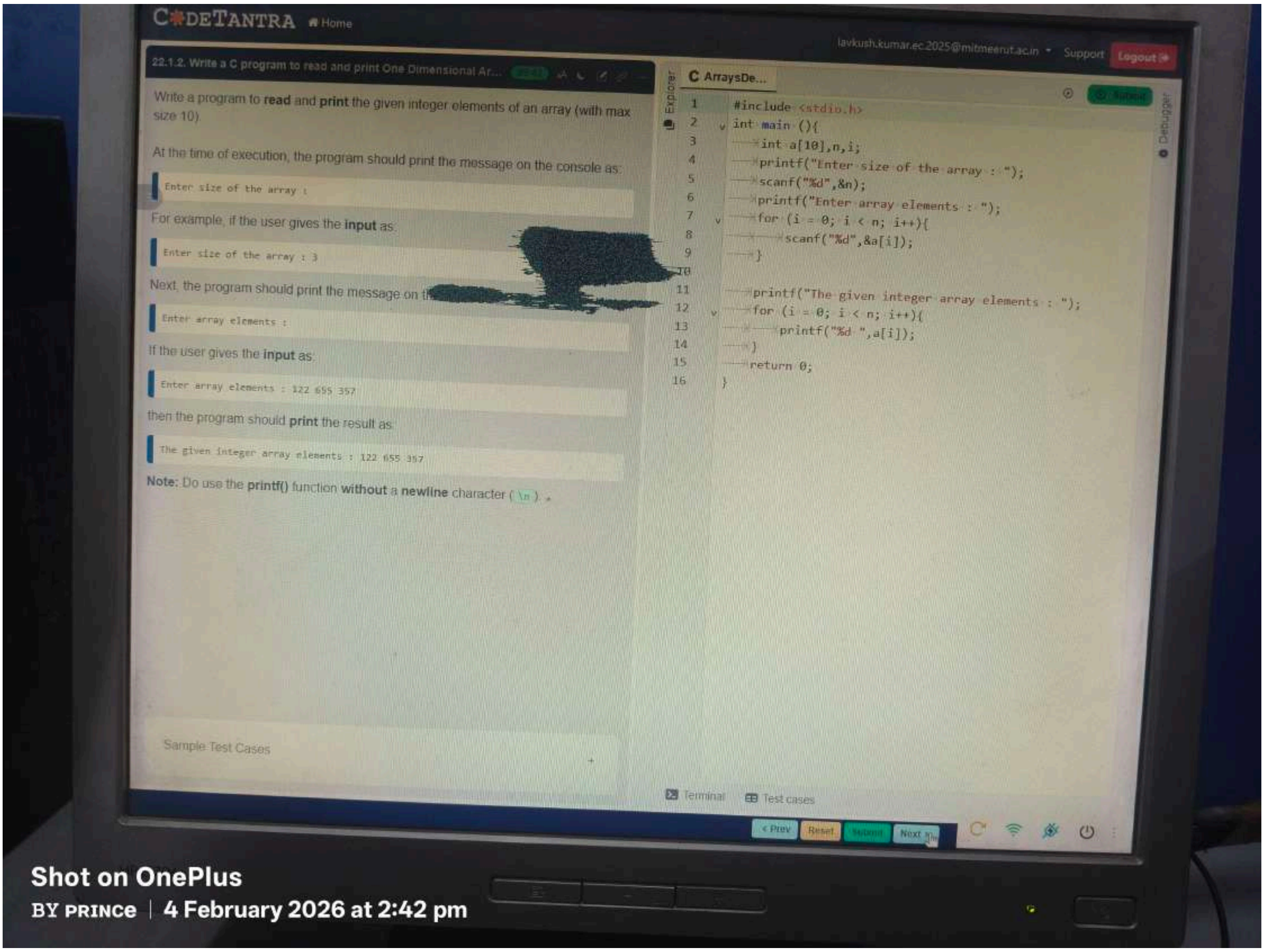
The formula for finding the i^{th} location of array element is

```
Address of the ith element = base_address + i * scale_factor
```

The **scale factor** is automatically calculated by the system, which is a value representing the size of the **data type** of the array.

Sample Test Cases

```
#include <stdio.h>
void main() {
	int a[10], n, i;
	printf("Enter how many values you want to read : ");
	scanf("%d", &n);
	for (i = 0; i < n; i++) {
		printf("Enter the value of a[%d] : ", i);
		scanf("%d", a + i);
	}
	printf("The array elements are : ");
	for (i = 0; i < n; i++) {
		printf(" %d ",*(a + i)); //Correct the code
	}
}
```



22.1.2. Write a C program to read and print One Dimensional Ar...

Write a program to **read** and **print** the given integer elements of an array (with max size 10).

At the time of execution, the program should print the message on the console as:

```
Enter size of the array :
```

For example, if the user gives the **input** as:

```
Enter size of the array : 3
```

Next, the program should print the message on th[redacted]

```
Enter array elements :
```

If the user gives the **input** as:

```
Enter array elements : 122 655 357
```

then the program should **print** the result as:

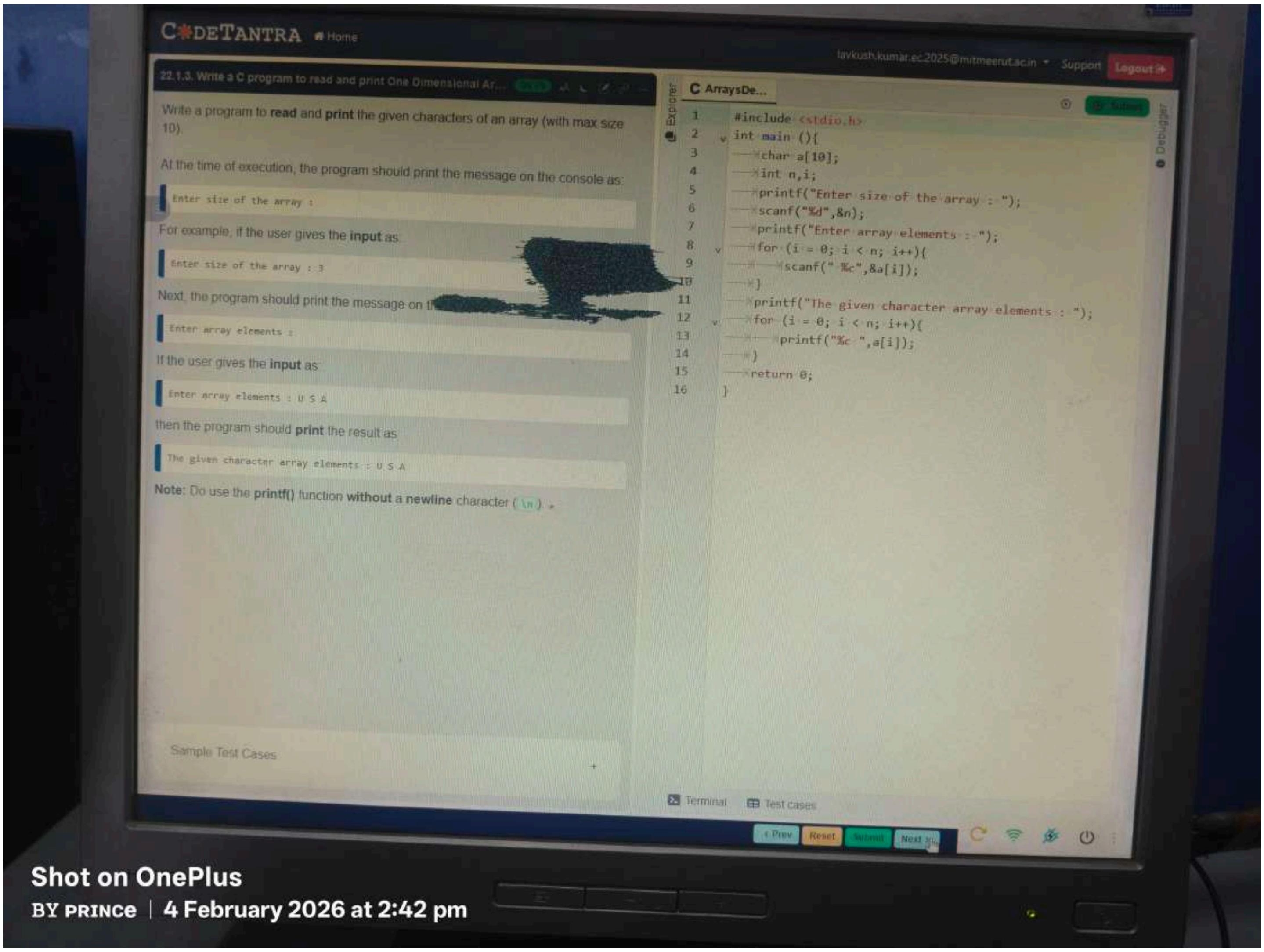
```
The given integer array elements : 122 655 357
```

Note: Do use the **printf()** function **without** a **newline** character (`\n`).

Sample Test Cases

C ArraysDe...

```
#include <stdio.h>
int main (){
	int a[10],n,i;
	printf("Enter size of the array : ");
	scanf("%d",&n);
	printf("Enter array elements : ");
	for (i = 0; i < n; i++){
		scanf("%d",&a[i]);
	}

	printf("The given integer array elements : ");
	for (i = 0; i < n; i++){
		printf("%d ",a[i]);
	}
	return 0;
}
```



22.1.3. Write a C program to read and print One Dimensional Ar...

Write a program to **read** and **print** the given characters of an array (with max size 10).

At the time of execution, the program should print the message on the console as:

```
Enter size of the array :
```

For example, if the user gives the **input** as:

```
Enter size of the array : 3
```

Next, the program should print the message on t[redacted]

```
Enter array elements :
```

If the user gives the **input** as:

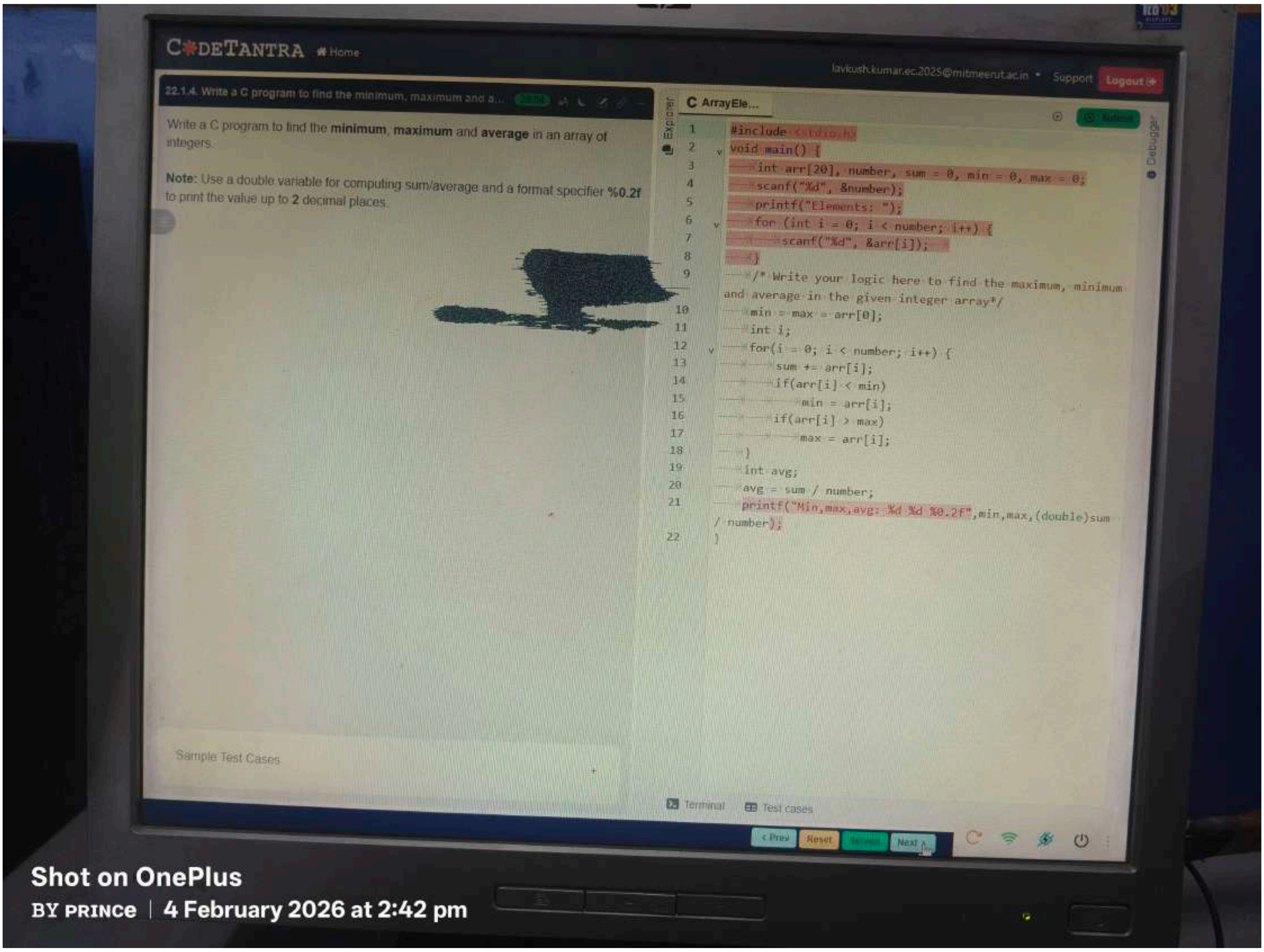
```
Enter array elements : U S A
```

then the program should **print** the result as:

```
The given character array elements : U S A
```

Note: Do use the **printf()** function **without** a **newline** character (`\n`).

```c
#include <stdio.h>
int main (){
	char a[10];
	int n,i;
	printf("Enter size of the array : ");
	scanf("%d",&n);
	printf("Enter array elements : ");
	for (i = 0; i < n; i++){
		scanf(" %c",&a[i]);
	}
	printf("The given character array elements : ");
	for (i = 0; i < n; i++){
		printf("%c ",a[i]);
	}
	return 0;
}
```



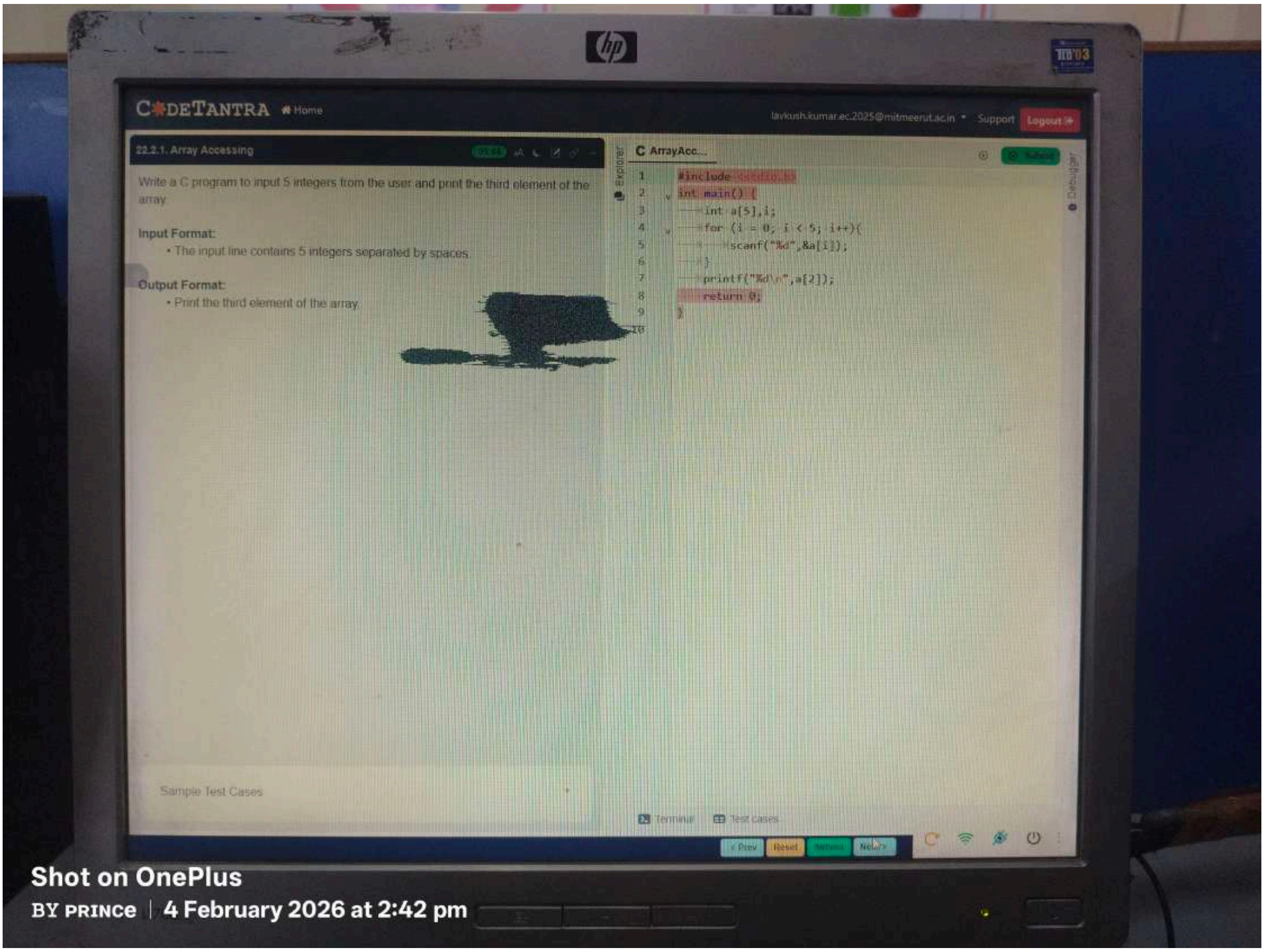
22.1.4. Write a C program to find the minimum, maximum and a...

Write a C program to find the **minimum**, **maximum** and **average** in an array of integers.

Note: Use a double variable for computing sum/average and a format specifier **%0.2f** to print the value up to **2** decimal places.

Sample Test Cases

```
#include <stdio.h>
void main() {
	int arr[20], number, sum = 0, min = 0, max = 0;
	scanf("%d", &number);
	printf("Elements: ");
	for (int i = 0; i < number; i++) {
		scanf("%d", &arr[i]);
	}
	/* Write your logic here to find the maximum, minimum and average in the given integer array*/
	min = max = arr[0];
	int i;
	for(i = 0; i < number; i++) {
		sum += arr[i];
		if(arr[i] < min)
			min = arr[i];
		if(arr[i] > max)
			max = arr[i];
	}
	int avg;
	avg = sum / number;
	printf("Min,max,avg: %d %d %0.2f",min,max,(double)sum / number);
}
```



22.2.1. Array Accessing

Write a C program to input 5 integers from the user and print the third element of the array

Input Format:

- The input line contains 5 integers separated by spaces.

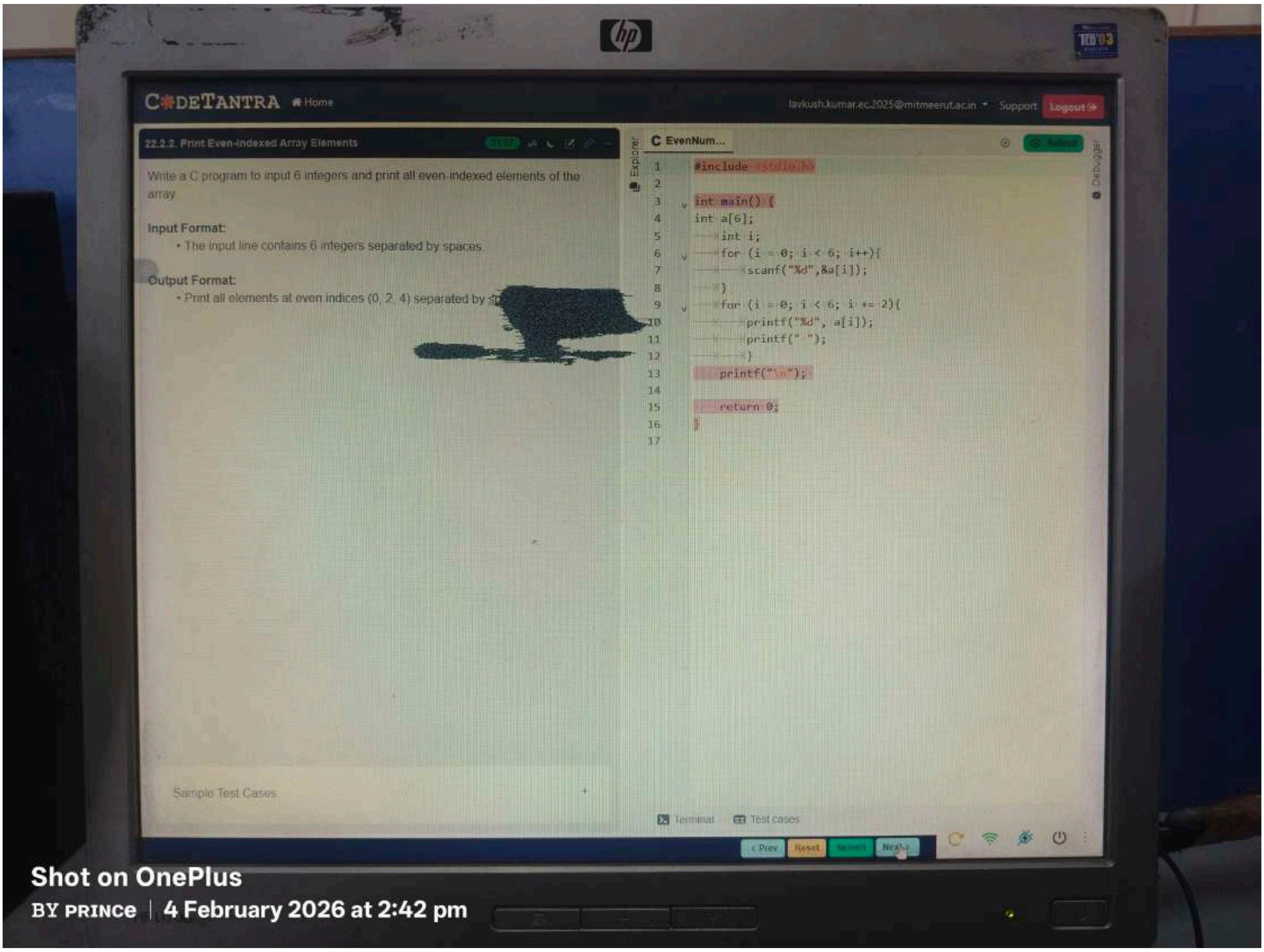
Output Format:

- Print the third element of the array.

Sample Test Cases

C ArrayAcc...

```c
#include <stdio.h>
int main() {
    int a[5],i;
    for (i = 0; i < 5; i++){
        scanf("%d",&a[i]);
    }
    printf("%d\n",a[2]);
    return 0;
}
```



22.2.2. Print Even-Indexed Array Elements

Write a C program to input 6 integers and print all even-indexed elements of the array

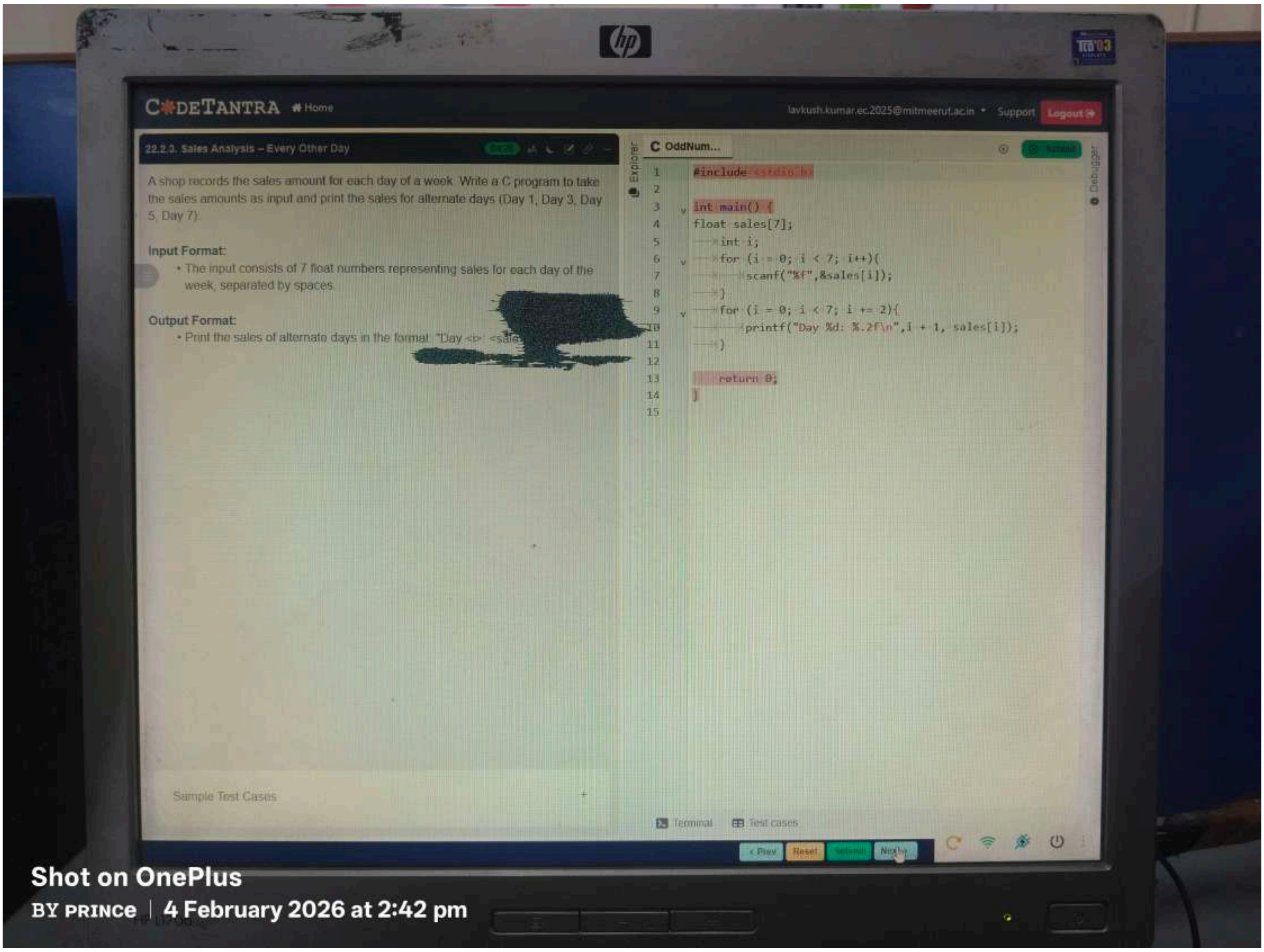
Input Format:

- The input line contains 6 integers separated by spaces

Output Format:

- Print all elements at even indices (0, 2, 4) separated by sp

```c
#include <stdio.h>

int main() {
int a[6];
	int i;
	for (i = 0; i < 6; i++){
		scanf("%d",&a[i]);
	}
	for (i = 0; i < 6; i += 2){
		printf("%d", a[i]);
		printf(" ");
		}
	printf("\n");

	return 0;
}
```



22.2.3. Sales Analysis – Every Other Day

A shop records the sales amount for each day of a week. Write a C program to take the sales amounts as input and print the sales for alternate days (Day 1, Day 3, Day 5, Day 7).

Input Format:

- The input consists of 7 float numbers representing sales for each day of the week, separated by spaces.

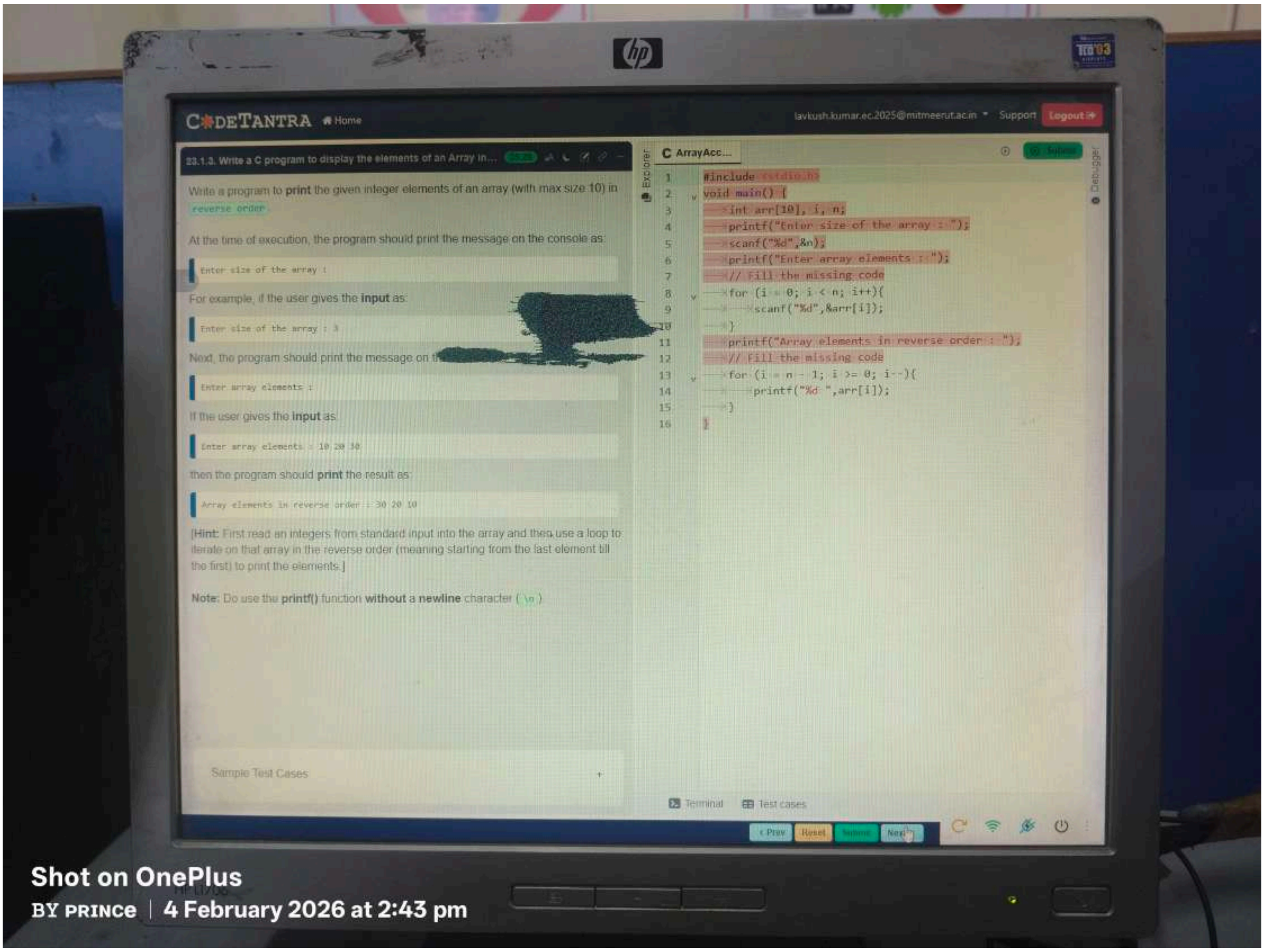
Output Format:

- Print the sales of alternate days in the format: "Day <i>: <sale [redacted]

Sample Test Cases

```c
#include <stdio.h>

int main() {
float sales[7];
	int i;
	for (i = 0; i < 7; i++){
		scanf("%f",&sales[i]);
	}
	for (i = 0; i < 7; i += 2){
		printf("Day %d: %.2f\n",i + 1, sales[i]);
	}

	return 0;
}
```

Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:42 pm



23.1.3. Write a C program to display the elements of an Array in...

Write a program to **print** the given integer elements of an array (with max size 10) in `reverse order`.

At the time of execution, the program should print the message on the console as:

```
Enter size of the array :
```

For example, if the user gives the **input** as: [redacted]

```
Enter size of the array : 3
```

Next, the program should print the message on th

```
Enter array elements :
```

If the user gives the **input** as:

```
Enter array elements : 10 20 30
```

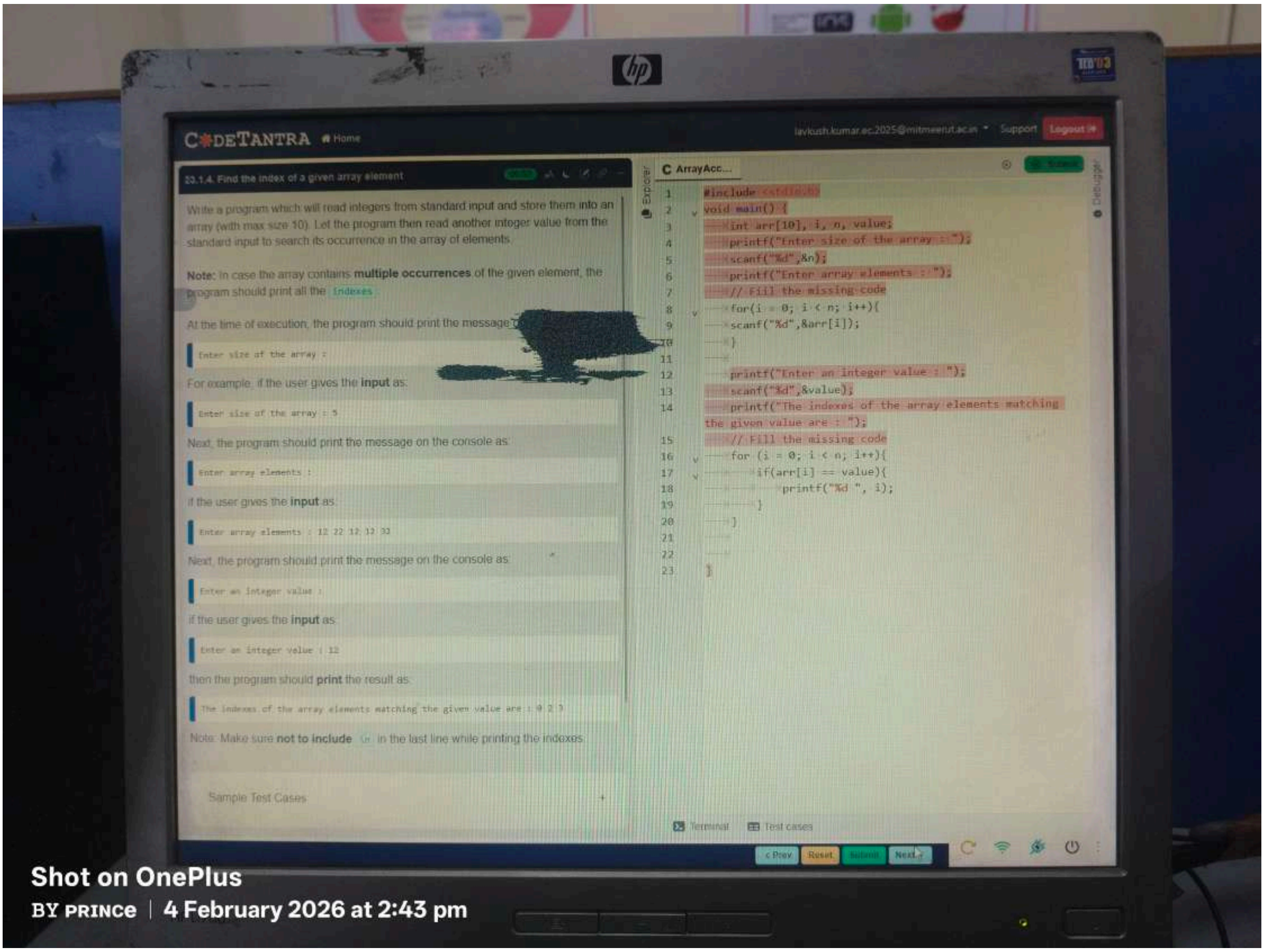
then the program should **print** the result as:

```
Array elements in reverse order : 30 20 10
```

[**Hint:** First read an integers from standard input into the array and then use a loop to iterate on that array in the reverse order (meaning starting from the last element till the first) to print the elements.]

Note: Do use the **printf()** function **without** a **newline** character (`\n`).

```c
#include <stdio.h>
void main() {
	int arr[10], i, n;
	printf("Enter size of the array : ");
	scanf("%d",&n);
	printf("Enter array elements : ");
	// Fill the missing code
	for (i = 0; i < n; i++){
		scanf("%d",&arr[i]);
	}
	printf("Array elements in reverse order : ");
	// Fill the missing code
	for (i = n - 1; i >= 0; i--){
		printf("%d ",arr[i]);
	}
}
```



23.1.4. Find the index of a given array element

Write a program which will read integers from standard input and store them into an array (with max size 10). Let the program then read another integer value from the standard input to search its occurrence in the array of elements.

Note: In case the array contains **multiple occurrences** of the given element, the program should print all the `indexes`.

At the time of execution, the program should print the message [redacted]

```
Enter size of the array :
```

For example, if the user gives the **input** as:

```
Enter size of the array : 5
```

Next, the program should print the message on the console as:

```
Enter array elements :
```

If the user gives the **input** as:

```
Enter array elements : 12 22 12 12 33
```

Next, the program should print the message on the console as:

```
Enter an integer value :
```

If the user gives the **input** as:

```
Enter an integer value : 12
```

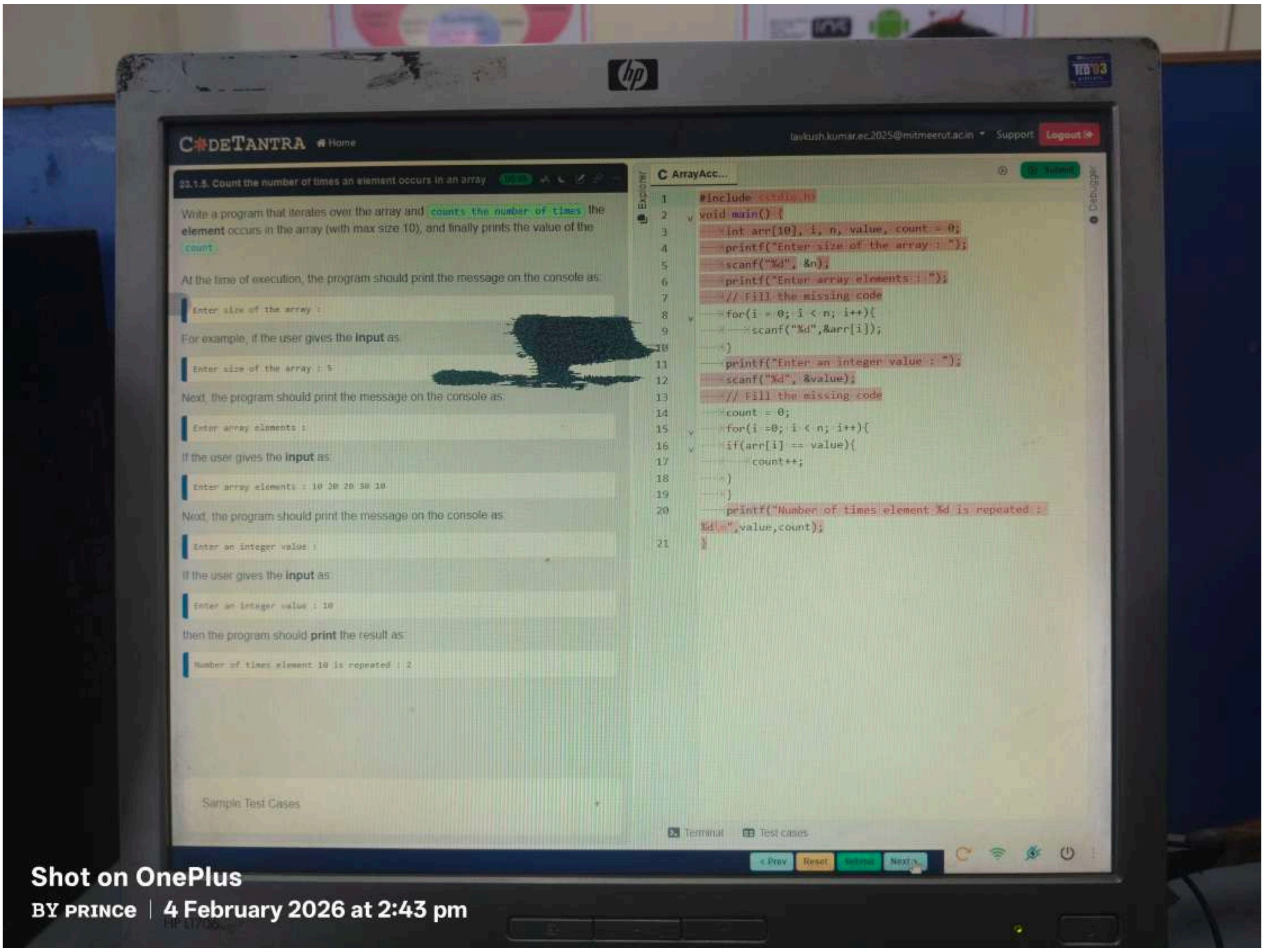
then the program should **print** the result as:

```
The indexes of the array elements matching the given value are : 0 2 3
```

Note: Make sure **not to include** `\n` in the last line while printing the indexes.

Sample Test Cases

```c
#include <stdio.h>
void main() {
	int arr[10], i, n, value;
	printf("Enter size of the array : ");
	scanf("%d",&n);
	printf("Enter array elements : ");
	// Fill the missing code
	for(i = 0; i < n; i++){
	scanf("%d",&arr[i]);
	}
	
	printf("Enter an integer value : ");
	scanf("%d",&value);
	printf("The indexes of the array elements matching the given value are : ");
	// Fill the missing code
	for (i = 0; i < n; i++){
		if(arr[i] == value){
			printf("%d ", i);
		}
	}
	
	
}
```



23.1.5. Count the number of times an element occurs in an array

Write a program that iterates over the array and `counts the number of times` the **element** occurs in the array (with max size 10), and finally prints the value of the `count`

At the time of execution, the program should print the message on the console as:

```
Enter size of the array :
```

For example, if the user gives the **input** as:

```
Enter size of the array : 5
```

[redacted]

Next, the program should print the message on the console as:

```
Enter array elements :
```

If the user gives the **input** as:

```
Enter array elements : 10 20 20 30 10
```

Next, the program should print the message on the console as:

```
Enter an integer value :
```

If the user gives the **input** as:

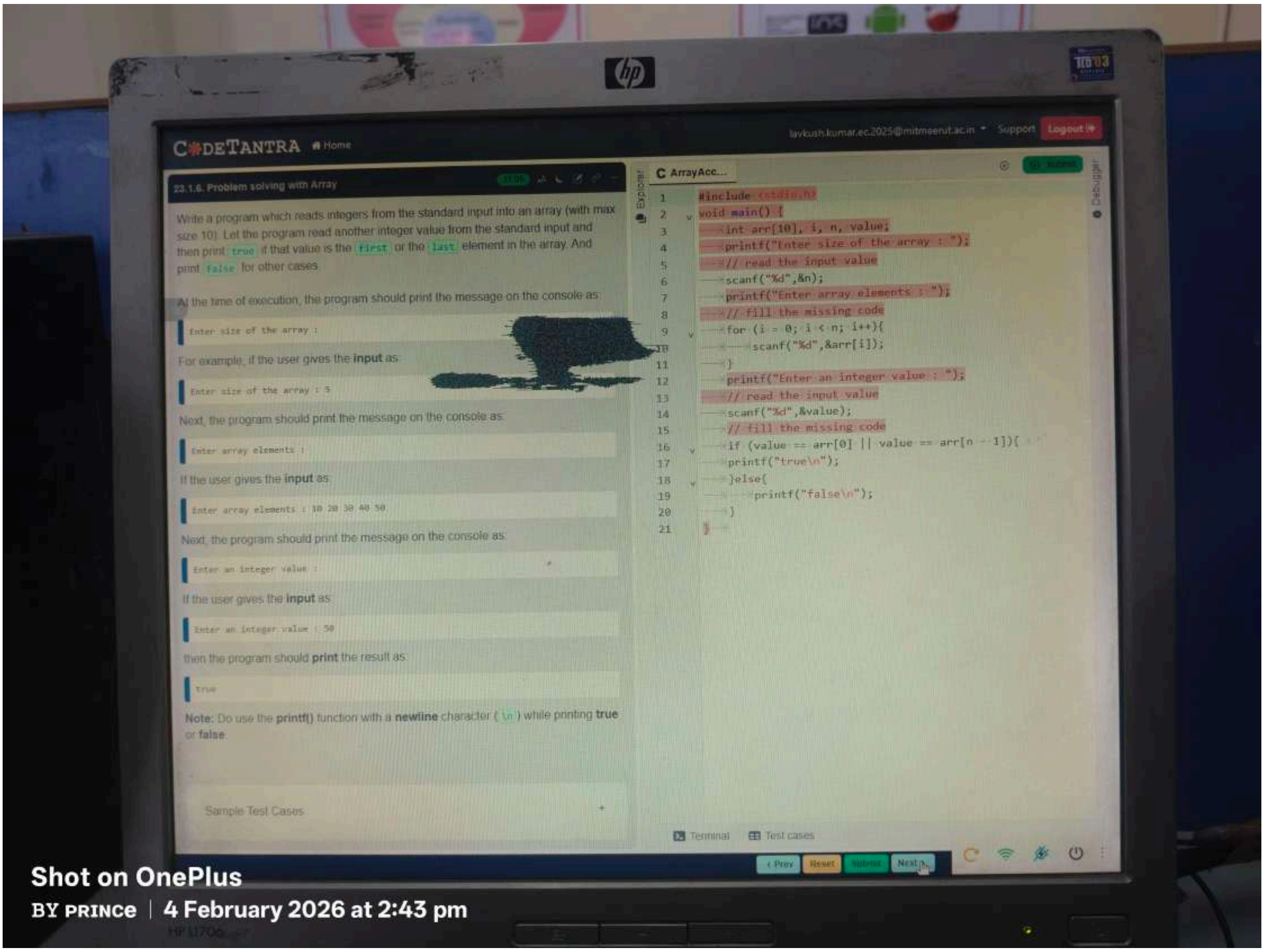
```
Enter an integer value : 10
```

then the program should **print** the result as:

```
Number of times element 10 is repeated : 2
```

ArrayAcc...

```c
#include <stdio.h>
void main() {
	int arr[10], i, n, value, count = 0;
	printf("Enter size of the array : ");
	scanf("%d", &n);
	printf("Enter array elements : ");
	// Fill the missing code
	for(i = 0; i < n; i++){
		scanf("%d",&arr[i]);
	}
	printf("Enter an integer value : ");
	scanf("%d", &value);
	// Fill the missing code
	count = 0;
	for(i =0; i < n; i++){
	if(arr[i] == value){
		count++;
	}
	}
	printf("Number of times element %d is repeated : %d\n",value,count);
}
```



23.1.6. Problem solving with Array

Write a program which reads integers from the standard input into an array (with max size 10). Let the program read another integer value from the standard input and then print `true` if that value is the `first` or the `last` element in the array. And print `false` for other cases.

At the time of execution, the program should print the message on the console as:

```
Enter size of the array :
```

[redacted]

For example, if the user gives the **input** as:

```
Enter size of the array : 5
```

Next, the program should print the message on the console as:

```
Enter array elements :
```

If the user gives the **input** as:

```
Enter array elements : 10 20 30 40 50
```

Next, the program should print the message on the console as:

```
Enter an integer value :
```

If the user gives the **input** as:

```
Enter an integer value : 50
```

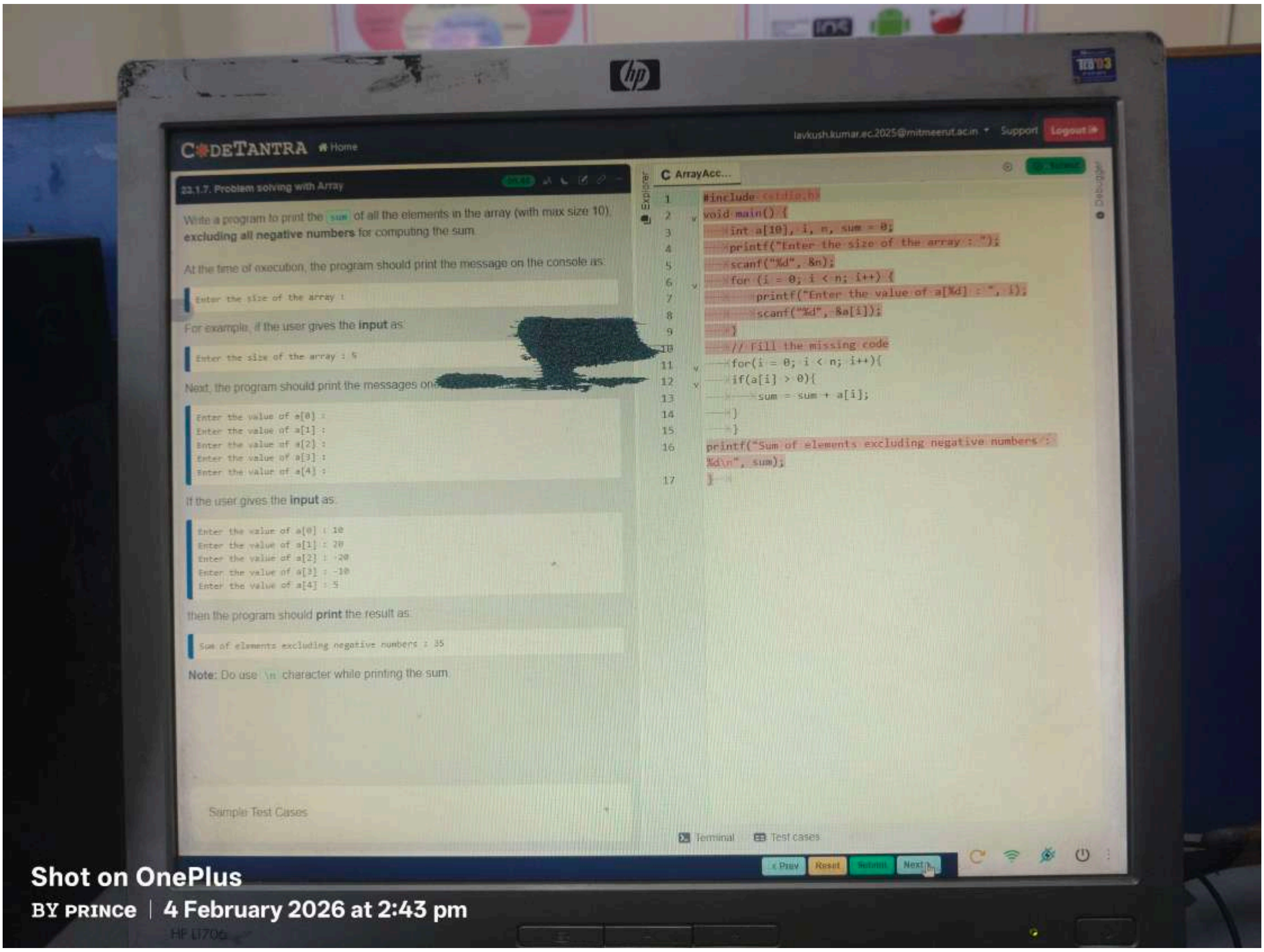
then the program should **print** the result as:

```
true
```

Note: Do use the **printf()** function with a **newline** character (`\n`) while printing **true** or **false**.

Sample Test Cases

```c
#include <stdio.h>
void main() {
	int arr[10], i, n, value;
	printf("Enter size of the array : ");
	// read the input value
	scanf("%d",&n);
	printf("Enter array elements : ");
	// fill the missing code
	for (i = 0; i < n; i++){
		scanf("%d",&arr[i]);
	}
	printf("Enter an integer value : ");
	// read the input value
	scanf("%d",&value);
	// fill the missing code
	if (value == arr[0] || value == arr[n - 1]){
	printf("true\n");
	}else{
		printf("false\n");
	}
}
```



23.1.7. Problem solving with Array

Write a program to print the `sum` of all the elements in the array (with max size 10) **excluding all negative numbers** for computing the sum.

At the time of execution, the program should print the message on the console as:

```
Enter the size of the array :
```

For example, if the user gives the **input** as: [redacted]

```
Enter the size of the array : 5
```

Next, the program should print the messages one [redacted]

```
Enter the value of a[0] :
Enter the value of a[1] :
Enter the value of a[2] :
Enter the value of a[3] :
Enter the value of a[4] :
```

If the user gives the **input** as:

```
Enter the value of a[0] : 10
Enter the value of a[1] : 20
Enter the value of a[2] : -20
Enter the value of a[3] : -10
Enter the value of a[4] : 5
```

then the program should **print** the result as:

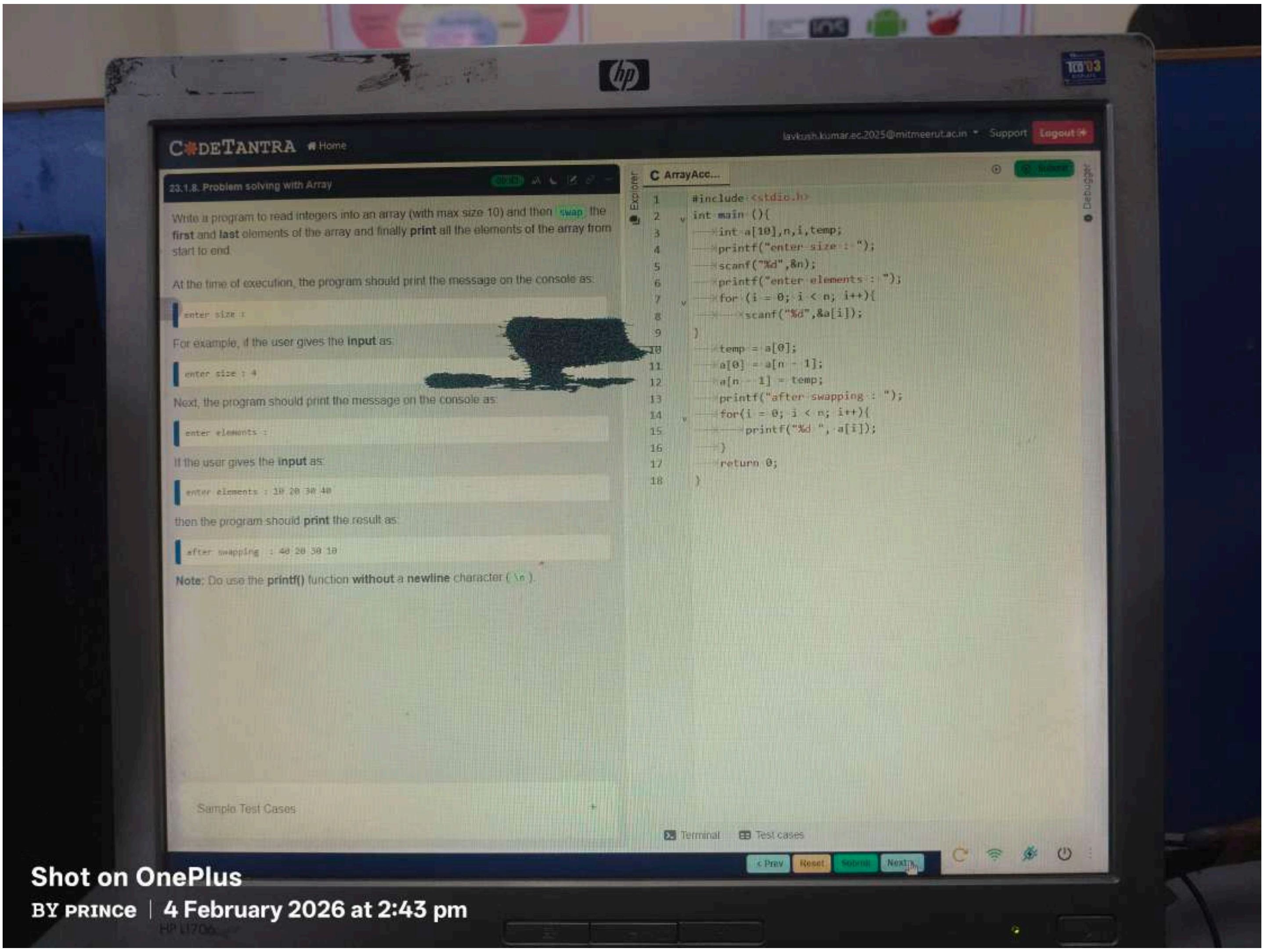
```
Sum of elements excluding negative numbers : 35
```

Note: Do use `\n` character while printing the sum.

Sample Test Cases

ArrayAcc...

```c
#include <stdio.h>
void main() {
	int a[10], i, n, sum = 0;
	printf("Enter the size of the array : ");
	scanf("%d", &n);
	for (i = 0; i < n; i++) {
		printf("Enter the value of a[%d] : ", i);
		scanf("%d", &a[i]);
	}
	// Fill the missing code
	for(i = 0; i < n; i++){
	if(a[i] > 0){
		sum = sum + a[i];
	}
	}
printf("Sum of elements excluding negative numbers : %d\n", sum);
}
```



23.1.8. Problem solving with Array

Write a program to read integers into an array (with max size 10) and then `swap` the **first** and **last** elements of the array and finally **print** all the elements of the array from start to end

At the time of execution, the program should print the message on the console as:

```
enter size :
```

For example, if the user gives the **input** as: [redacted]

```
enter size : 4
```

Next, the program should print the message on the console as:

```
enter elements :
```

If the user gives the **input** as:

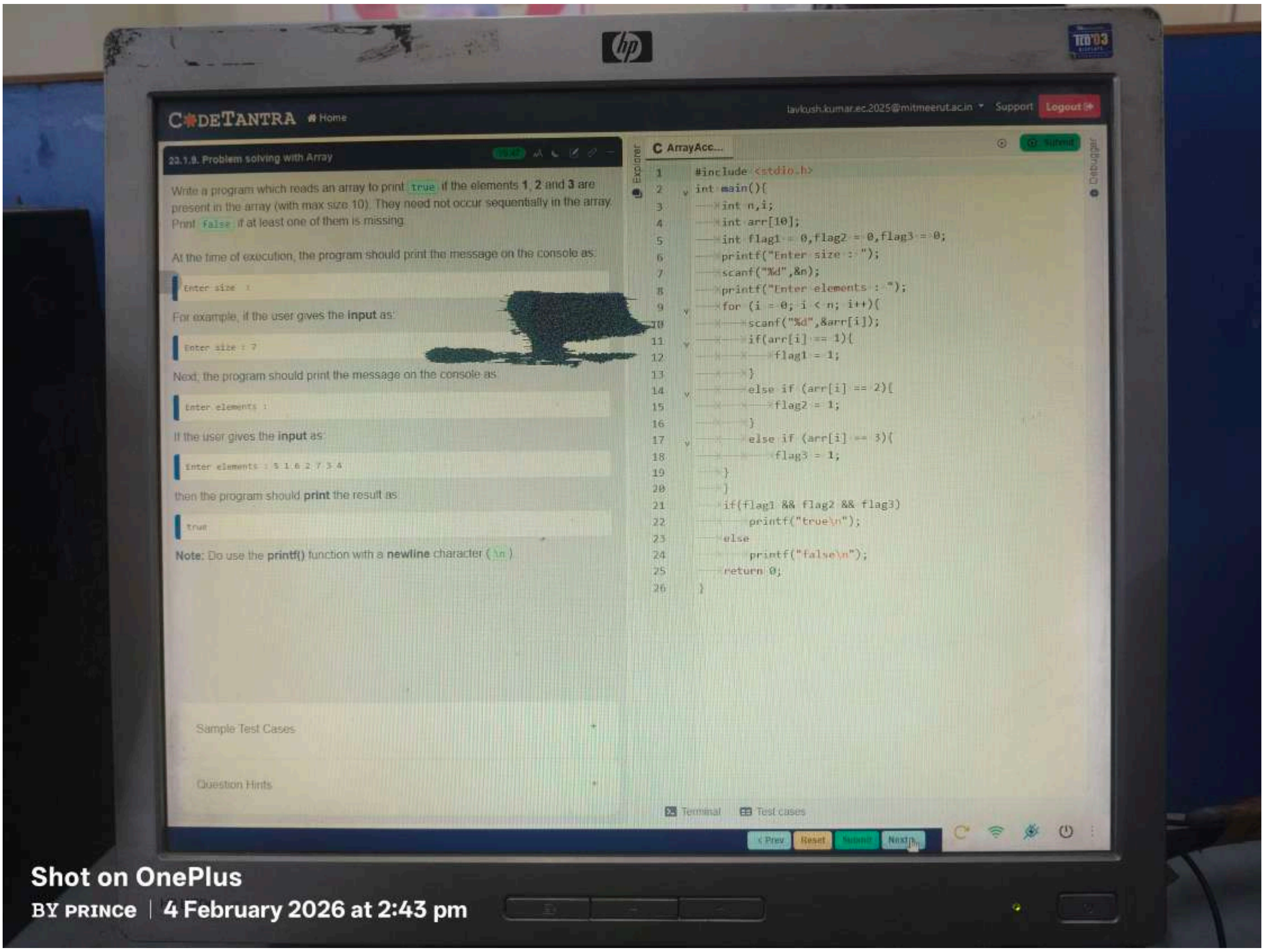
```
enter elements : 10 20 30 40
```

then the program should **print** the result as:

```
after swapping : 40 20 30 10
```

Note: Do use the **printf()** function **without** a **newline** character (`\n`).

```c
#include <stdio.h>
int main (){
	int a[10],n,i,temp;
	printf("enter size : ");
	scanf("%d",&n);
	printf("enter elements : ");
	for (i = 0; i < n; i++){
		scanf("%d",&a[i]);
}
	temp = a[0];
	a[0] = a[n - 1];
	a[n - 1] = temp;
	printf("after swapping : ");
	for(i = 0; i < n; i++){
		printf("%d ", a[i]);
	}
	return 0;
}
```



23.1.9. Problem solving with Array

Write a program which reads an array to print `true` if the elements **1**, **2** and **3** are present in the array (with max size 10). They need not occur sequentially in the array. Print `false` if at least one of them is missing.

At the time of execution, the program should print the message on the console as:

```
Enter size :
```

For example, if the user gives the **input** as:

[redacted]

```
Enter size : 7
```

Next, the program should print the message on the console as:

```
Enter elements :
```

If the user gives the **input** as:

```
Enter elements : 5 1 6 2 7 3 4
```

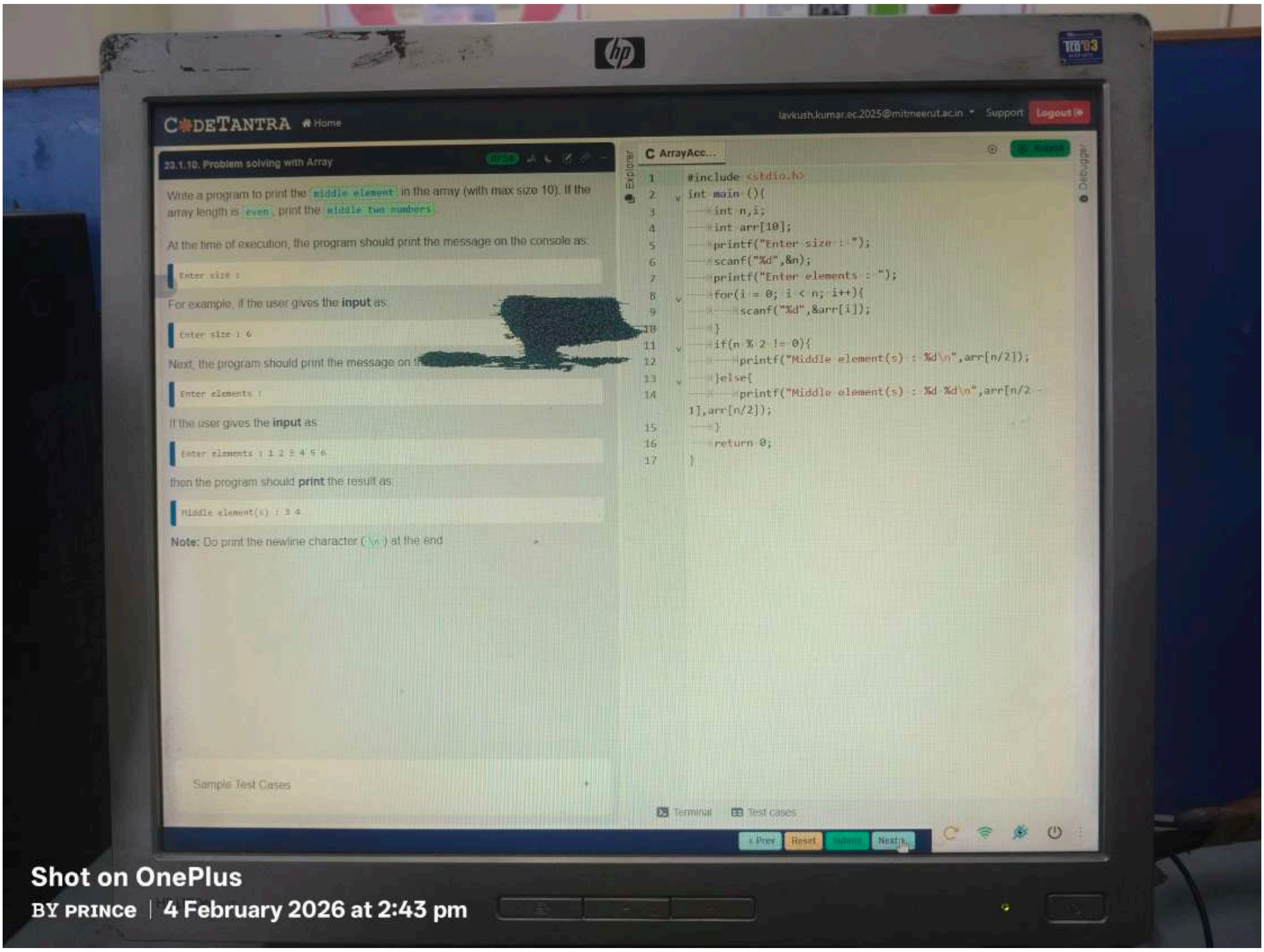
then the program should **print** the result as:

```
true
```

Note: Do use the **printf()** function with a **newline** character (`\n`).

C ArrayAcc...

```c
#include <stdio.h>
int main(){
	int n,i;
	int arr[10];
	int flag1 = 0,flag2 = 0,flag3 = 0;
	printf("Enter size : ");
	scanf("%d",&n);
	printf("Enter elements : ");
	for (i = 0; i < n; i++){
		scanf("%d",&arr[i]);
		if(arr[i] == 1){
			flag1 = 1;
		}
		else if (arr[i] == 2){
			flag2 = 1;
		}
		else if (arr[i] == 3){
			flag3 = 1;
		}
	}
	if(flag1 && flag2 && flag3)
		printf("true\n");
	else
		printf("false\n");
	return 0;
}
```



23.1.10. Problem solving with Array

Write a program to print the `middle element` in the array (with max size 10). If the array length is `even`, print the `middle two numbers`.

At the time of execution, the program should print the message on the console as:

```
Enter size :
```

For example, if the user gives the **input** as: [redacted]

```
Enter size : 6
```

Next, the program should print the message on th[redacted]

```
Enter elements :
```

If the user gives the **input** as:

```
Enter elements : 1 2 3 4 5 6
```

then the program should **print** the result as:

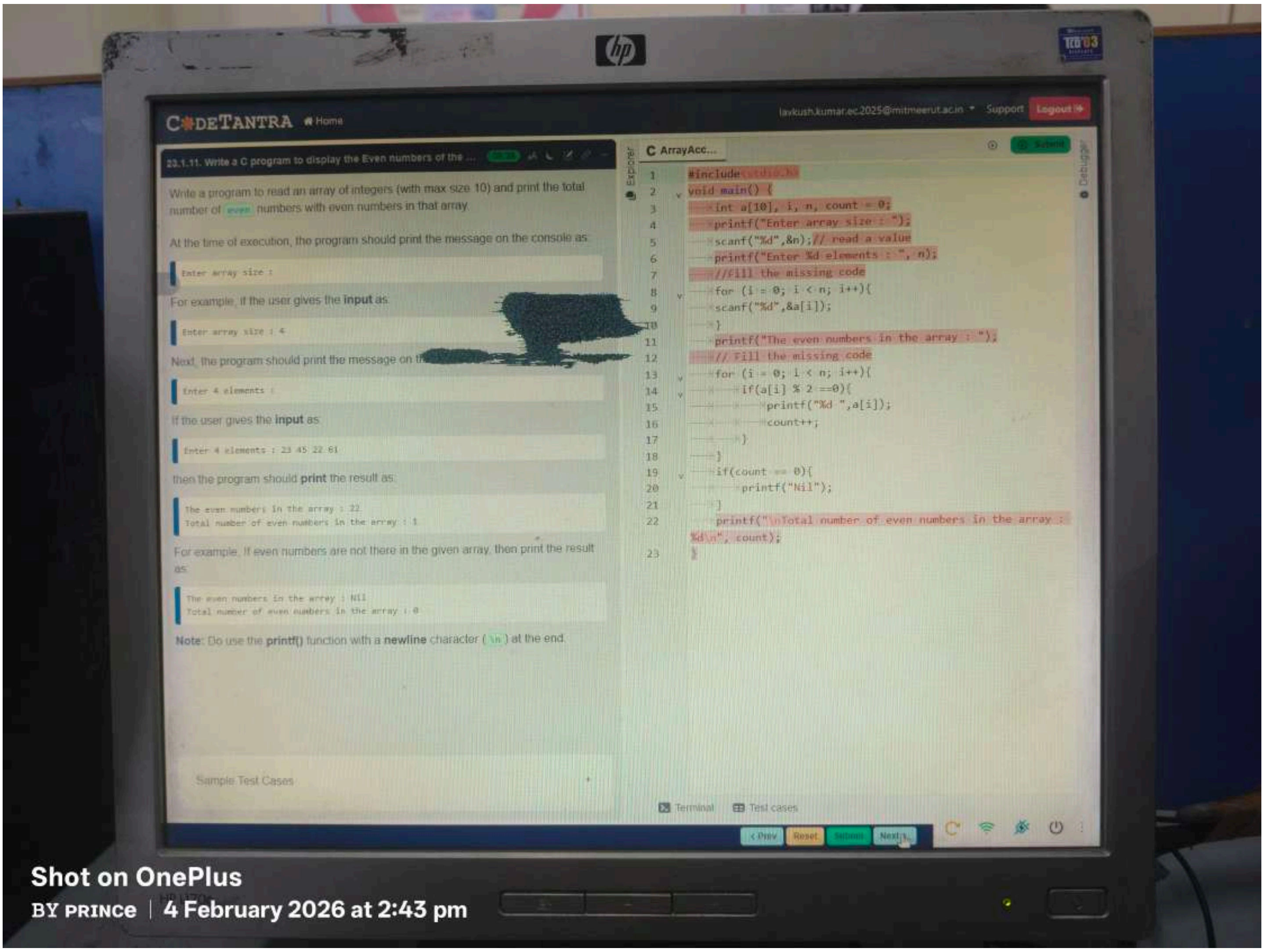
```
Middle element(s) : 3 4
```

Note: Do print the newline character (`\n`) at the end

Sample Test Cases

C ArrayAcc...

```c
#include <stdio.h>
int main (){
	int n,i;
	int arr[10];
	printf("Enter size : ");
	scanf("%d",&n);
	printf("Enter elements : ");
	for(i = 0; i < n; i++){
		scanf("%d",&arr[i]);
	}
	if(n % 2 != 0){
		printf("Middle element(s) : %d\n",arr[n/2]);
	}else{
		printf("Middle element(s) : %d %d\n",arr[n/2 - 1],arr[n/2]);
	}
	return 0;
}
```



23.1.11. Write a C program to display the Even numbers of the ...

Write a program to read an array of integers (with max size 10) and print the total number of `even` numbers with even numbers in that array.

At the time of execution, the program should print the message on the console as:

```
Enter array size :
```

For example, if the user gives the **input** as: [redacted]

```
Enter array size : 4
```

Next, the program should print the message on th[redacted]

```
Enter 4 elements :
```

If the user gives the **input** as:

```
Enter 4 elements : 23 45 22 61
```

then the program should **print** the result as:

```
The even numbers in the array : 22
Total number of even numbers in the array : 1
```

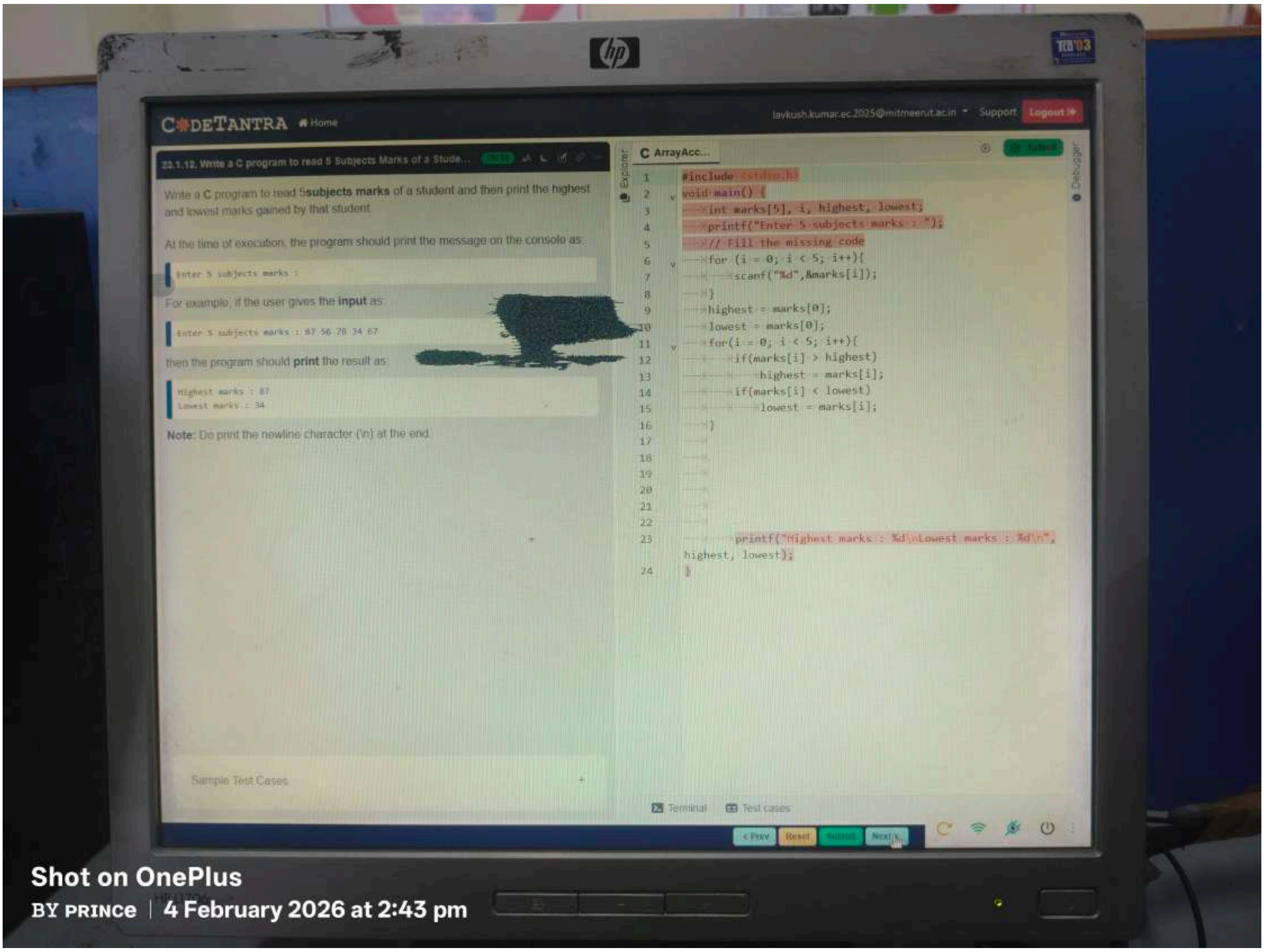
For example, if even numbers are not there in the given array, then print the result as:

```
The even numbers in the array : Nil
Total number of even numbers in the array : 0
```

Note: Do use the **printf()** function with a **newline** character (`\n`) at the end.

C ArrayAcc...

```c
#include<stdio.h>
void main() {
	int a[10], i, n, count = 0;
	printf("Enter array size : ");
	scanf("%d",&n);// read a value
	printf("Enter %d elements : ", n);
	//Fill the missing code
	for (i = 0; i < n; i++){
	scanf("%d",&a[i]);
	}
	printf("The even numbers in the array : ");
	// Fill the missing code
	for (i = 0; i < n; i++){
		if(a[i] % 2 ==0){
			printf("%d ",a[i]);
			count++;
		}
	}
	if(count == 0){
		printf("Nil");
	}
	printf("\nTotal number of even numbers in the array : %d\n", count);
}
```



23.1.12. Write a C program to read 5 Subjects Marks of a Stude...

Write a **C** program to read 5**subjects marks** of a student and then print the highest and lowest marks gained by that student.

At the time of execution, the program should print the message on the console as:

```
Enter 5 subjects marks :
```

For example, if the user gives the **input** as:

[redacted]

```
Enter 5 subjects marks : 87 56 78 34 67
```

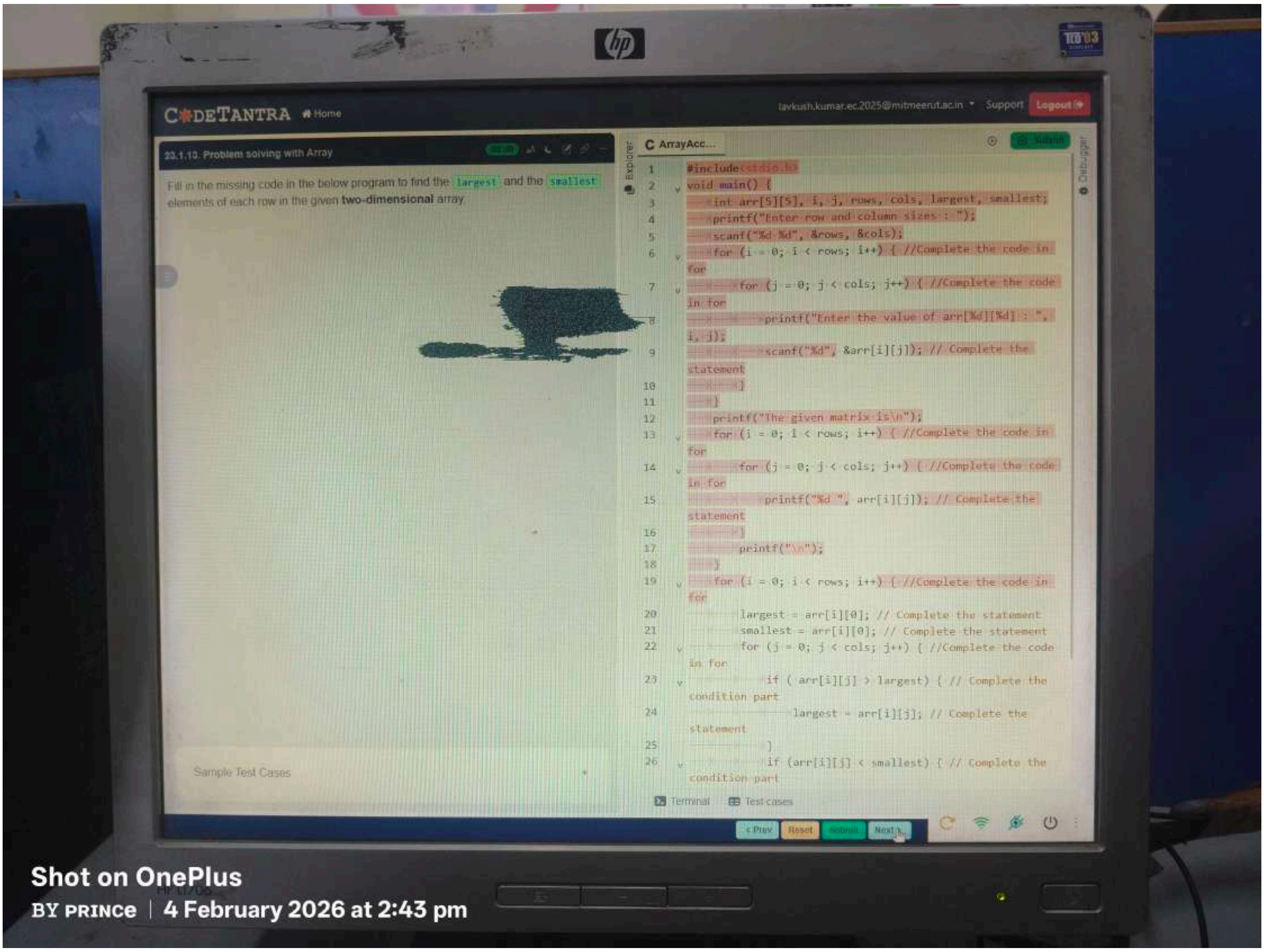
then the program should **print** the result as:

```
Highest marks : 87
Lowest marks : 34
```

Note: Do print the newline character (\n) at the end.

```c
#include <stdio.h>
void main() {
	int marks[5], i, highest, lowest;
	printf("Enter 5 subjects marks : ");
	// Fill the missing code
	for (i = 0; i < 5; i++){
		scanf("%d",&marks[i]);
	}
	highest = marks[0];
	lowest = marks[0];
	for(i = 0; i < 5; i++){
		if(marks[i] > highest)
			highest = marks[i];
		if(marks[i] < lowest)
			lowest = marks[i];
	}
	
	
	
	
	
	
		printf("Highest marks : %d\nLowest marks : %d\n", highest, lowest);
}
```

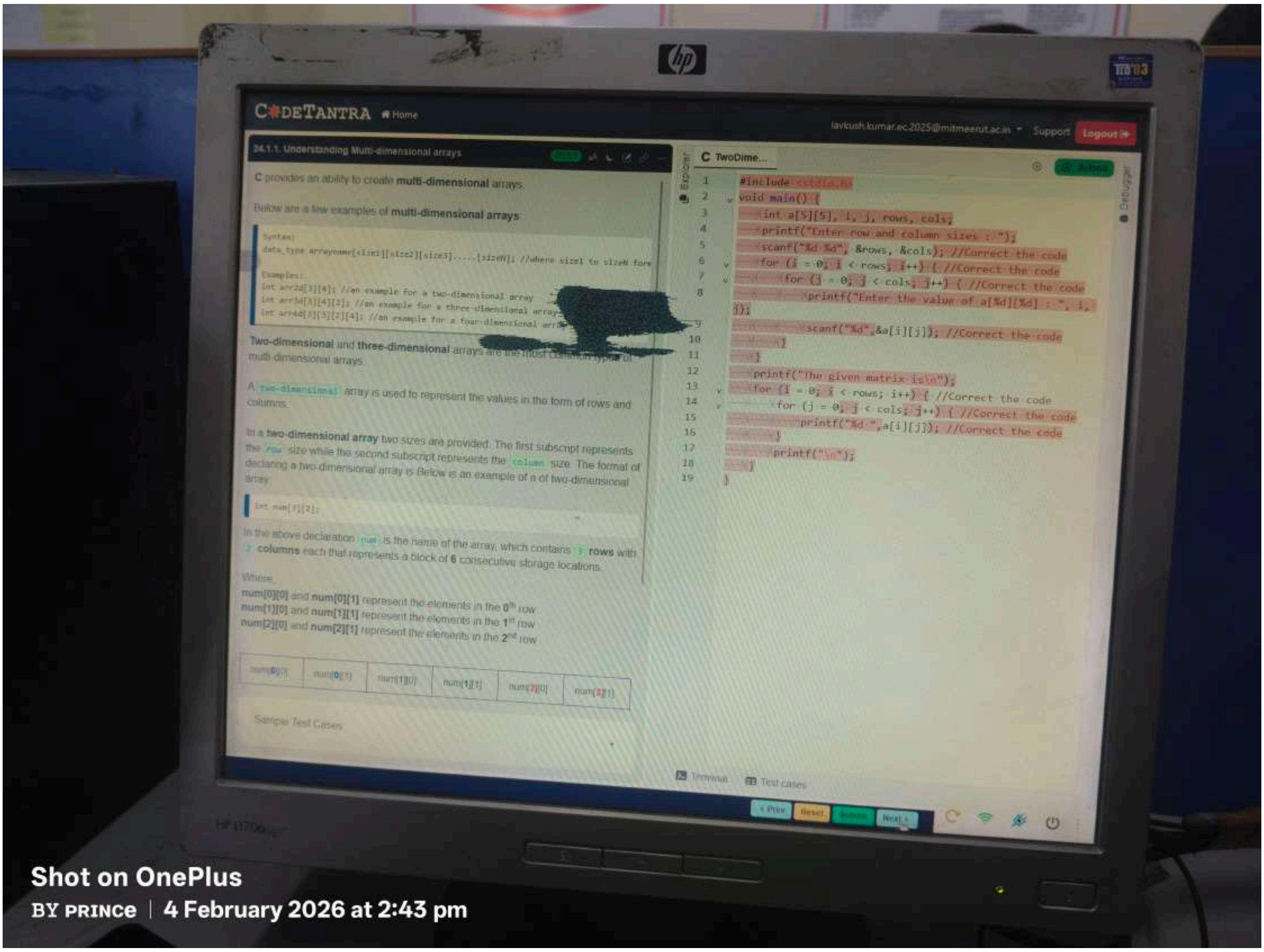
Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:43 pm



23.1.13. Problem solving with Array

Fill in the missing code in the below program to find the `largest` and the `smallest` elements of each row in the given **two-dimensional** array.

```c
#include<stdio.h>
void main() {
	int arr[5][5], i, j, rows, cols, largest, smallest;
	printf("Enter row and column sizes : ");
	scanf("%d %d", &rows, &cols);
	for (i = 0; i < rows; i++) { //Complete the code in for
		for (j = 0; j < cols; j++) { //Complete the code in for
			printf("Enter the value of arr[%d][%d] : ", i, j);
			scanf("%d", &arr[i][j]); // Complete the statement
		}
	}
	printf("The given matrix is\n");
	for (i = 0; i < rows; i++) { //Complete the code in for
		for (j = 0; j < cols; j++) { //Complete the code in for
			printf("%d ", arr[i][j]); // Complete the statement
		}
		printf("\n");
	}
	for (i = 0; i < rows; i++) { //Complete the code in for
		largest = arr[i][0]; // Complete the statement
		smallest = arr[i][0]; // Complete the statement
		for (j = 0; j < cols; j++) { //Complete the code in for
			if ( arr[i][j] > largest) { // Complete the condition part
				largest = arr[i][j]; // Complete the statement
			}
			if (arr[i][j] < smallest) { // Complete the condition part
```



24.1.1. Understanding Multi-dimensional arrays

C provides an ability to create **multi-dimensional** arrays.

Below are a few examples of **multi-dimensional arrays**:

```
Syntax:
data_type arrayname[size1][size2][size3].....[sizeN]; //where size1 to sizeN form
Examples:
int arr2d[3][4]; //an example for a two-dimensional array
int arr3d[3][4][2]; //an example for a three-dimensional array
int arr4d[3][3][2][4]; //an example for a four-dimensional arr
```

[redacted]

Two-dimensional and **three-dimensional** arrays are the most common types of multi-dimensional arrays.

A `two-dimensional` array is used to represent the values in the form of rows and columns.

In a **two-dimensional array** two sizes are provided. The first subscript represents the `row` size while the second subscript represents the `column` size. The format of declaring a two-dimensional array is Below is an example of a of two-dimensional array.

```
int num[3][2];
```

In the above declaration `num` is the name of the array, which contains `3` **rows** with `2` **columns** each that represents a block of **6** consecutive storage locations.

Where,

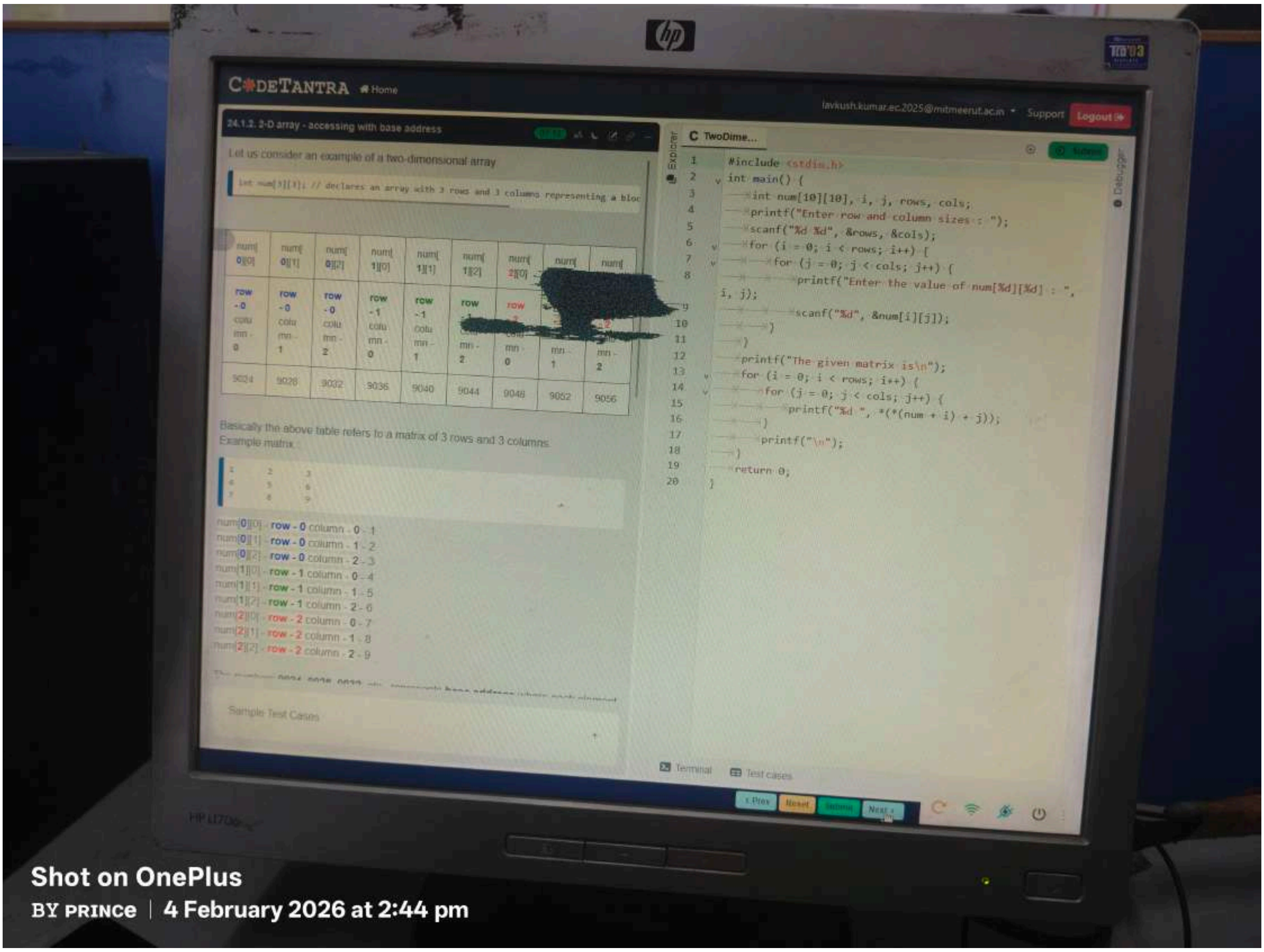
num[0][0] and **num[0][1]** represent the elements in the **0**th row
num[1][0] and **num[1][1]** represent the elements in the **1**st row
num[2][0] and **num[2][1]** represent the elements in the **2**nd row

num[0][0]	num[0][1]	num[1][0]	num[1][1]	num[2][0]	num[2][1]

Sample Test Cases

C TwoDime...

```
#include <stdio.h>
void main() {
	int a[5][5], i, j, rows, cols;
	printf("Enter row and column sizes : ");
	scanf("%d %d", &rows, &cols); //Correct the code
	for (i = 0; i < rows; i++) { //Correct the code
		for (j = 0; j < cols; j++) { //Correct the code
			printf("Enter the value of a[%d][%d] : ", i, j);
			scanf("%d",&a[i][j]); //Correct the code
		}
	}
	printf("The given matrix is\n");
	for (i = 0; i < rows; i++) { //Correct the code
		for (j = 0; j < cols; j++) { //Correct the code
			printf("%d ",a[i][j]); //Correct the code
		}
		printf("\n");
	}
}
```



24.1.2. 2-D array - accessing with base address

Let us consider an example of a two-dimensional array

```
int num[3][3]; // declares an array with 3 rows and 3 columns representing a bloc
```

num[0][0]	num[0][1]	num[0][2]	num[1][0]	num[1][1]	num[1][2]	num[2][0] [redacted]	num[[redacted]	num[[redacted]
row - 0 colu mn - 0	row - 0 colu mn - 1	row - 0 colu mn - 2	row - 1 colu mn - 0	row - 1 colu mn - 1	row [redacted] mn - 2	row [redacted] mn - 0	[redacted] mn - 1	[redacted] mn - 2
9024	9028	9032	9036	9040	9044	9048	9052	9056

Basically the above table refers to a matrix of 3 rows and 3 columns.
Example matrix :

```
1    2    3
4    5    6
7    8    9
```

num[0][0] - row - 0 column - 0 - 1
num[0][1] - row - 0 column - 1 - 2
num[0][2] - row - 0 column - 2 - 3
num[1][0] - row - 1 column - 0 - 4
num[1][1] - row - 1 column - 1 - 5
num[1][2] - row - 1 column - 2 - 6
num[2][0] - row - 2 column - 0 - 7
num[2][1] - row - 2 column - 1 - 8
num[2][2] - row - 2 column - 2 - 9

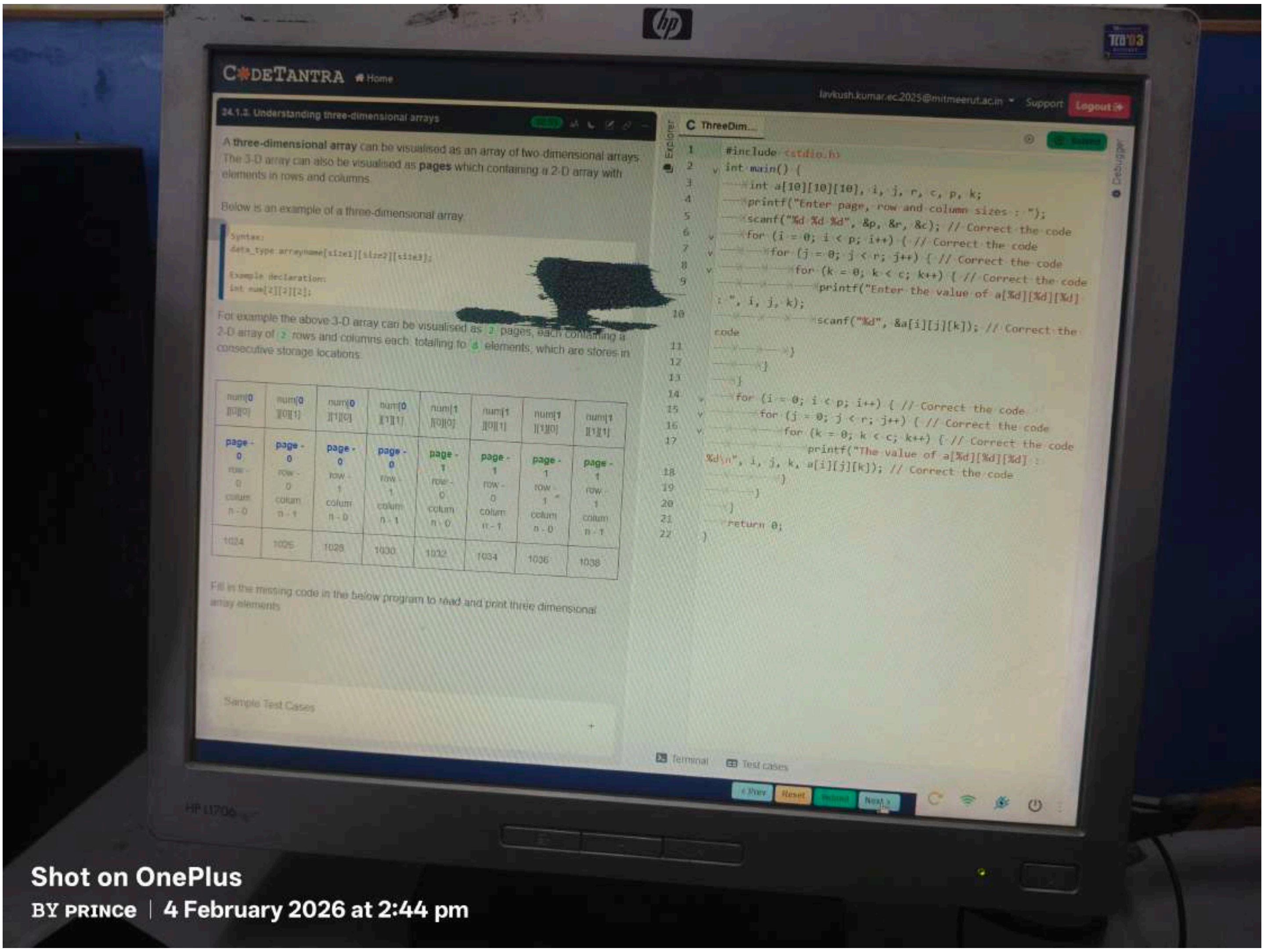
The numbers 9024, 9028, 9032 etc. represents base address [illegible]

Sample Test Cases

C TwoDime...

```c
#include <stdio.h>
int main() {
	int num[10][10], i, j, rows, cols;
	printf("Enter row and column sizes : ");
	scanf("%d %d", &rows, &cols);
	for (i = 0; i < rows; i++) {
		for (j = 0; j < cols; j++) {
			printf("Enter the value of num[%d][%d] : ", i, j);
			scanf("%d", &num[i][j]);
		}
	}
	printf("The given matrix is\n");
	for (i = 0; i < rows; i++) {
		for (j = 0; j < cols; j++) {
			printf("%d ", *(*(num + i) + j));
		}
		printf("\n");
	}
	return 0;
}
```

Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:44 pm



24.1.2. Understanding three-dimensional arrays

A **three-dimensional array** can be visualised as an array of two-dimensional arrays. The 3-D array can also be visualised as **pages** which containing a 2-D array with elements in rows and columns.

Below is an example of a three-dimensional array

```
Syntax:
data_type arrayname[size1][size2][size3];

Example declaration:
int num[2][2][2];
```

For example the above 3-D array can be visualised as 2 pages, each containing a 2-D array of 2 rows and columns each, totalling to 8 elements, which are stores in consecutive storage locations.

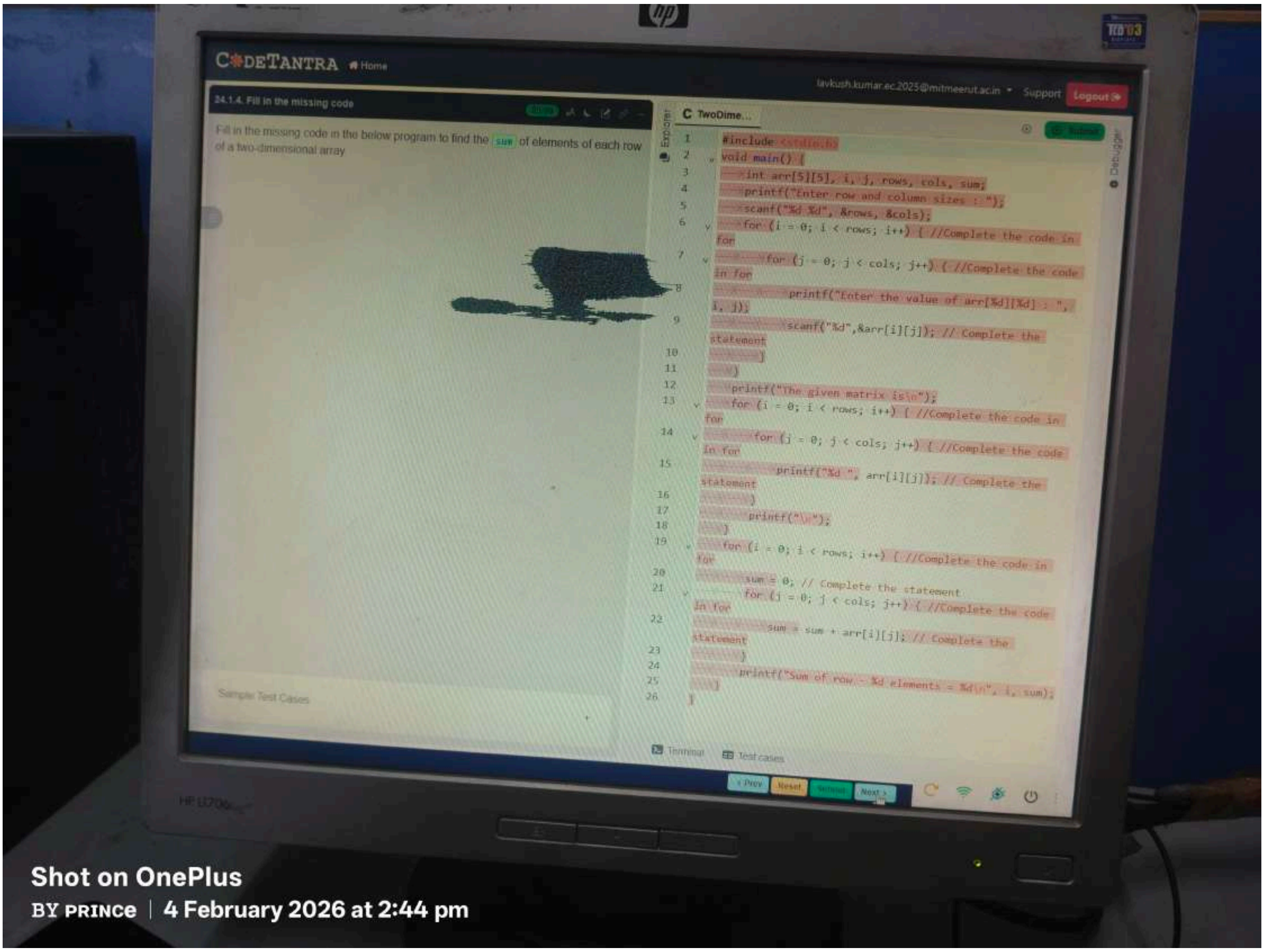
num[0][0][0]	num[0][0][1]	num[0][1][0]	num[0][1][1]	num[1][0][0]	num[1][0][1]	num[1][1][0]	num[1][1][1]
page - 0 row - 0 column - 0	page - 0 row - 0 column - 1	page - 0 row - 1 column - 0	page - 0 row - 1 column - 1	page - 1 row - 0 column - 0	page - 1 row - 0 column - 1	page - 1 row - 1 column - 0	page - 1 row - 1 column - 1
1024	1026	1028	1030	1032	1034	1036	1038

Fill in the missing code in the below program to read and print three dimensional array elements

Sample Test Cases

```c
#include <stdio.h>
int main() {
	int a[10][10][10], i, j, r, c, p, k;
	printf("Enter page, row and column sizes : ");
	scanf("%d %d %d", &p, &r, &c); // Correct the code
	for (i = 0; i < p; i++) { // Correct the code
		for (j = 0; j < r; j++) { // Correct the code
			for (k = 0; k < c; k++) { // Correct the code
				printf("Enter the value of a[%d][%d][%d] : ", i, j, k);
				scanf("%d", &a[i][j][k]); // Correct the code
			}
		}
	}
	for (i = 0; i < p; i++) { // Correct the code
		for (j = 0; j < r; j++) { // Correct the code
			for (k = 0; k < c; k++) { // Correct the code
				printf("The value of a[%d][%d][%d] : %d\n", i, j, k, a[i][j][k]); // Correct the code
			}
		}
	}
	return 0;
}
```

Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:44 pm

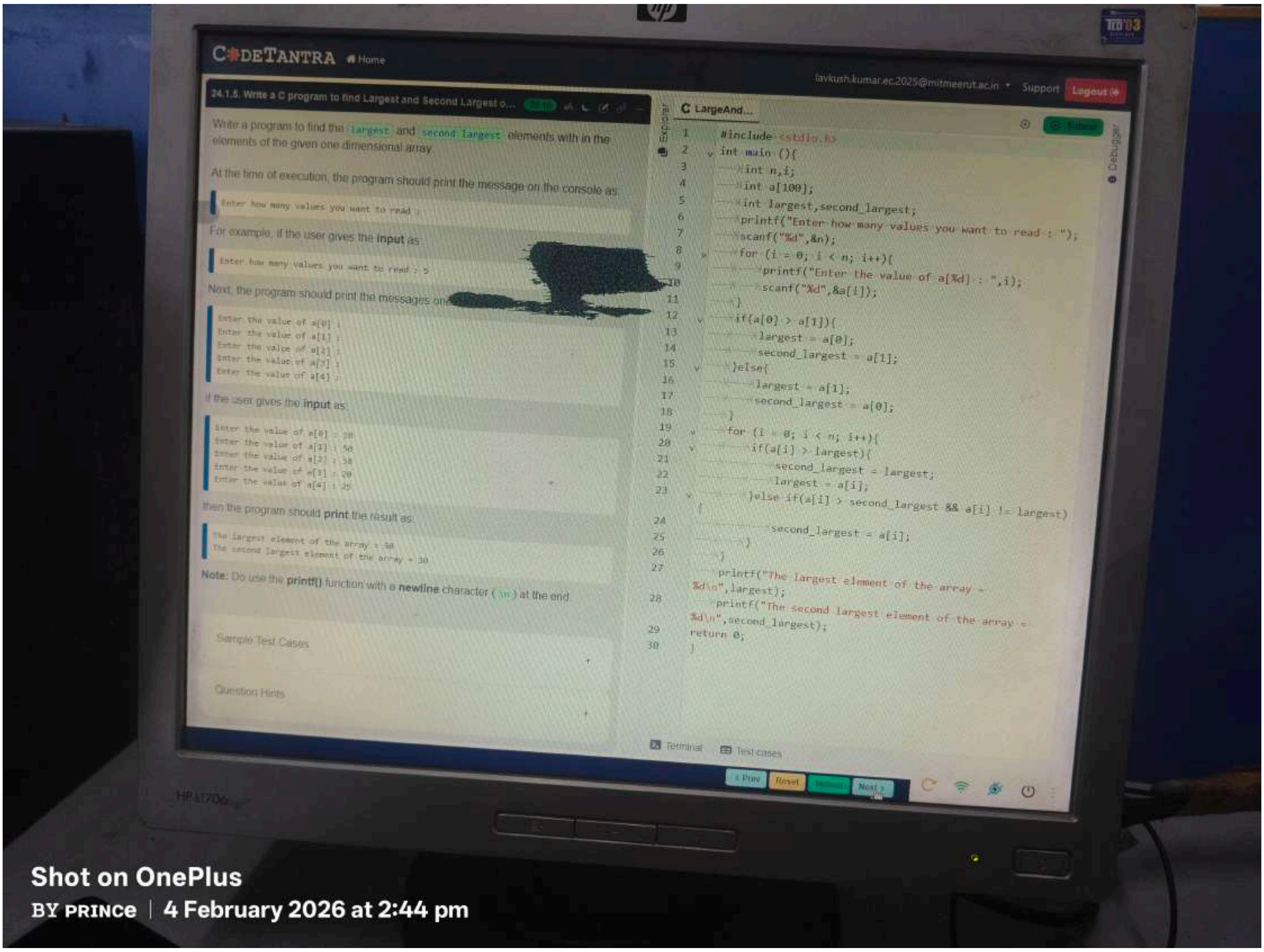


24.1.4. Fill in the missing code

Fill in the missing code in the below program to find the `sum` of elements of each row of a two-dimensional array

```c
#include <stdio.h>
void main() {
	int arr[5][5], i, j, rows, cols, sum;
	printf("Enter row and column sizes : ");
	scanf("%d %d", &rows, &cols);
	for (i = 0; i < rows; i++) { //Complete the code in for
		for (j = 0; j < cols; j++) { //Complete the code in for
			printf("Enter the value of arr[%d][%d] : ", i, j);
			scanf("%d",&arr[i][j]); // Complete the statement
		}
	}
	printf("The given matrix is\n");
	for (i = 0; i < rows; i++) { //Complete the code in for
		for (j = 0; j < cols; j++) { //Complete the code in for
			printf("%d ", arr[i][j]); // Complete the statement
		}
		printf("\n");
	}
	for (i = 0; i < rows; i++) { //Complete the code in for
		sum = 0; // Complete the statement
		for (j = 0; j < cols; j++) { //Complete the code in for
			sum = sum + arr[i][j]; // Complete the statement
		}
		printf("Sum of row - %d elements = %d\n", i, sum);
	}
}
```

Sample Test Cases



24.1.5. Write a C program to find Largest and Second Largest o...

Write a program to find the `largest` and `second largest` elements with in the elements of the given one dimensional array.

At the time of execution, the program should print the message on the console as:

```
Enter how many values you want to read :
```

For example, if the user gives the **input** as:

```
Enter how many values you want to read : 5
```

Next, the program should print the messages one [redacted]

```
Enter the value of a[0] :
Enter the value of a[1] :
Enter the value of a[2] :
Enter the value of a[3] :
Enter the value of a[4] :
```

If the user gives the **input** as:

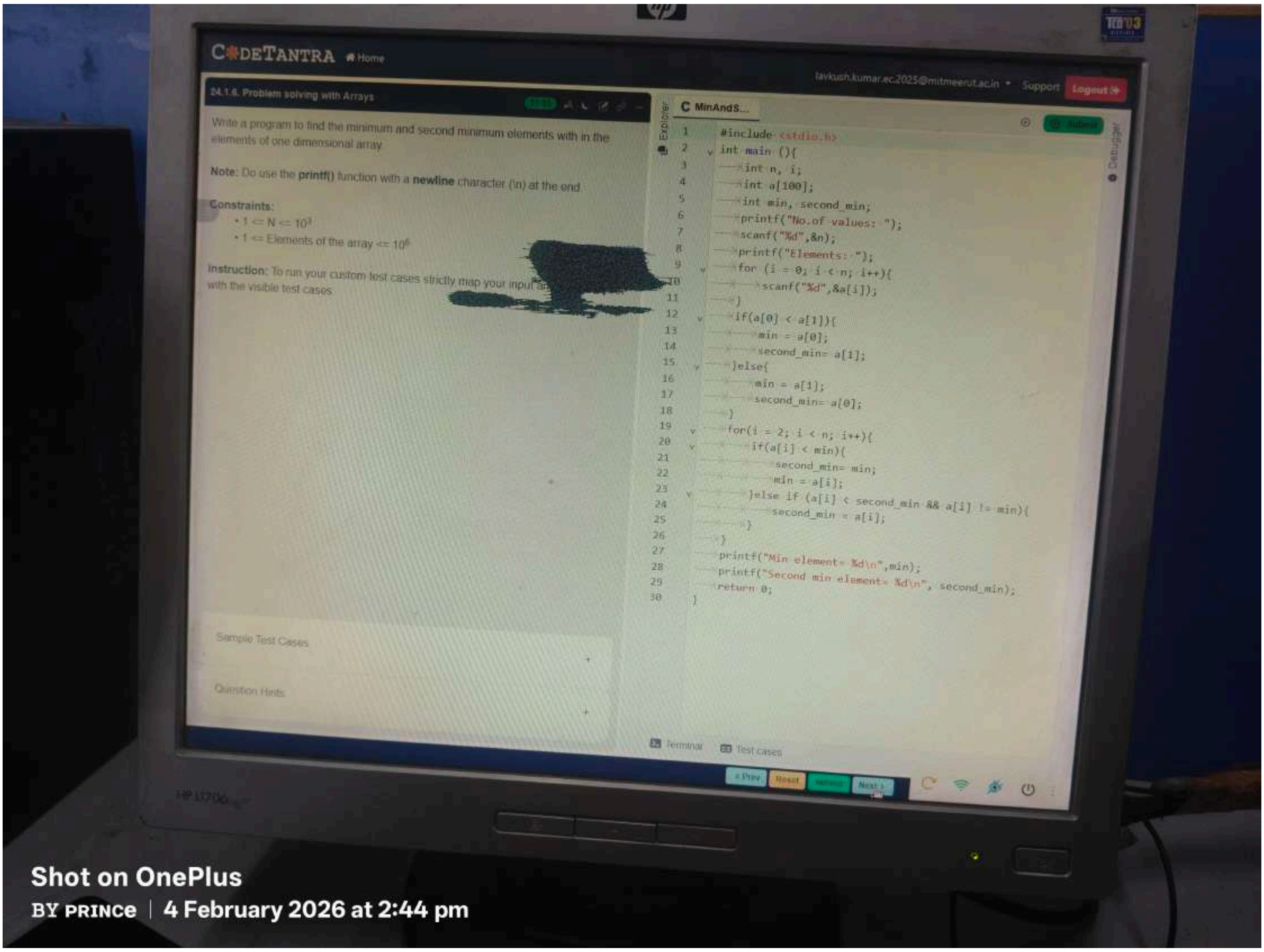
```
Enter the value of a[0] : 10
Enter the value of a[1] : 50
Enter the value of a[2] : 30
Enter the value of a[3] : 20
Enter the value of a[4] : 25
```

then the program should **print** the result as:

```
The largest element of the array = 50
The second largest element of the array = 30
```

Note: Do use the **printf()** function with a **newline** character (`\n`) at the end.

```c
#include <stdio.h>
int main (){
	int n,i;
	int a[100];
	int largest,second_largest;
	printf("Enter how many values you want to read : ");
	scanf("%d",&n);
	for (i = 0; i < n; i++){
		printf("Enter the value of a[%d] : ",i);
		scanf("%d",&a[i]);
	}
	if(a[0] > a[1]){
		largest = a[0];
		second_largest = a[1];
	}else{
		largest = a[1];
		second_largest = a[0];
	}
	for (i = 0; i < n; i++){
		if(a[i] > largest){
			second_largest = largest;
			largest = a[i];
		}else if(a[i] > second_largest && a[i] != largest){
			second_largest = a[i];
		}
	}
	printf("The largest elmment of the array = %d\n",largest);
	printf("The second largest element of the array = %d\n",second_largest);
return 0;
}
```



24.1.6. Problem solving with Arrays

Write a program to find the minimum and second minimum elements with in the elements of one dimensional array

Note: Do use the **printf()** function with a **newline** character (\n) at the end

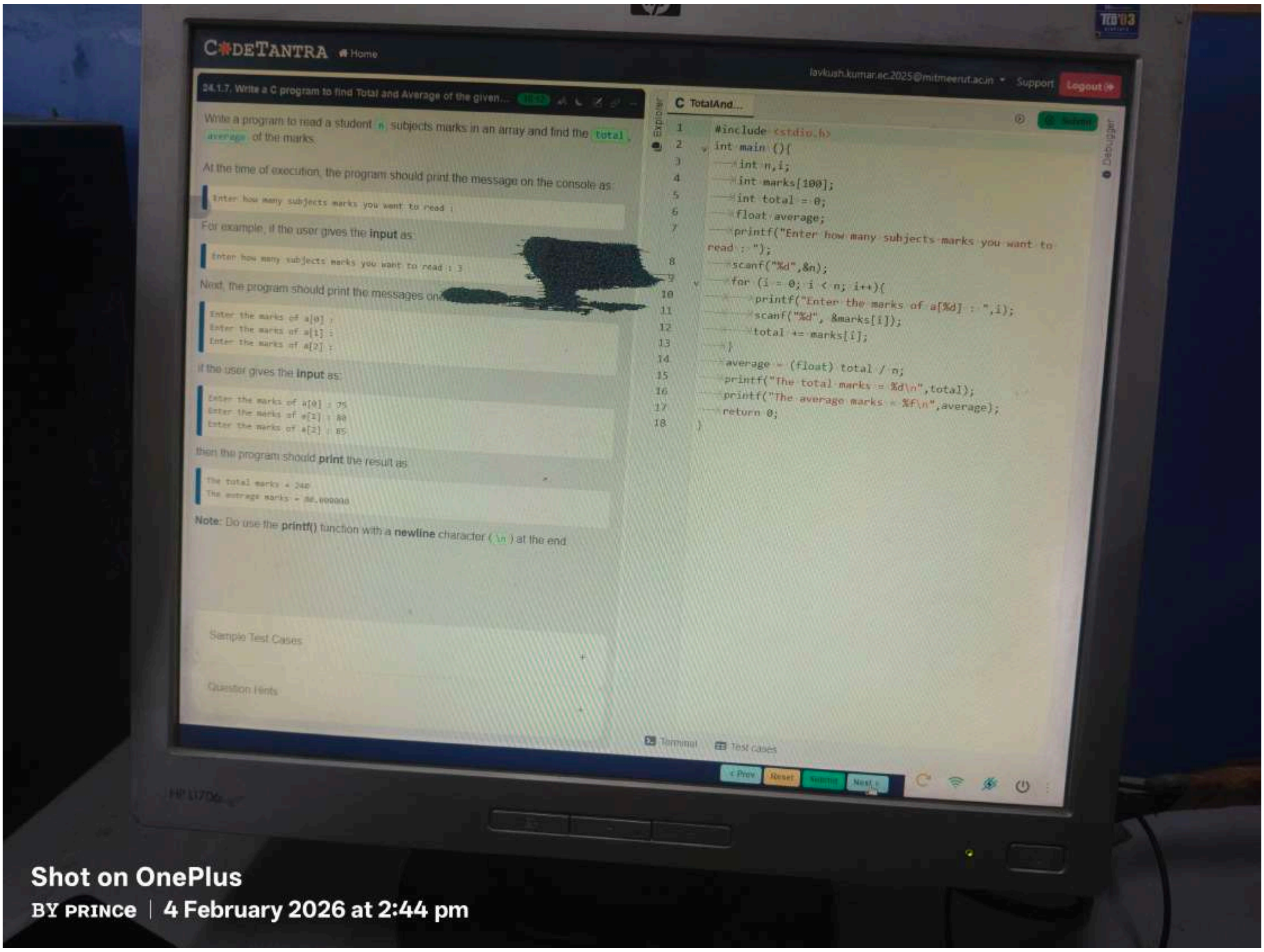
Constraints:

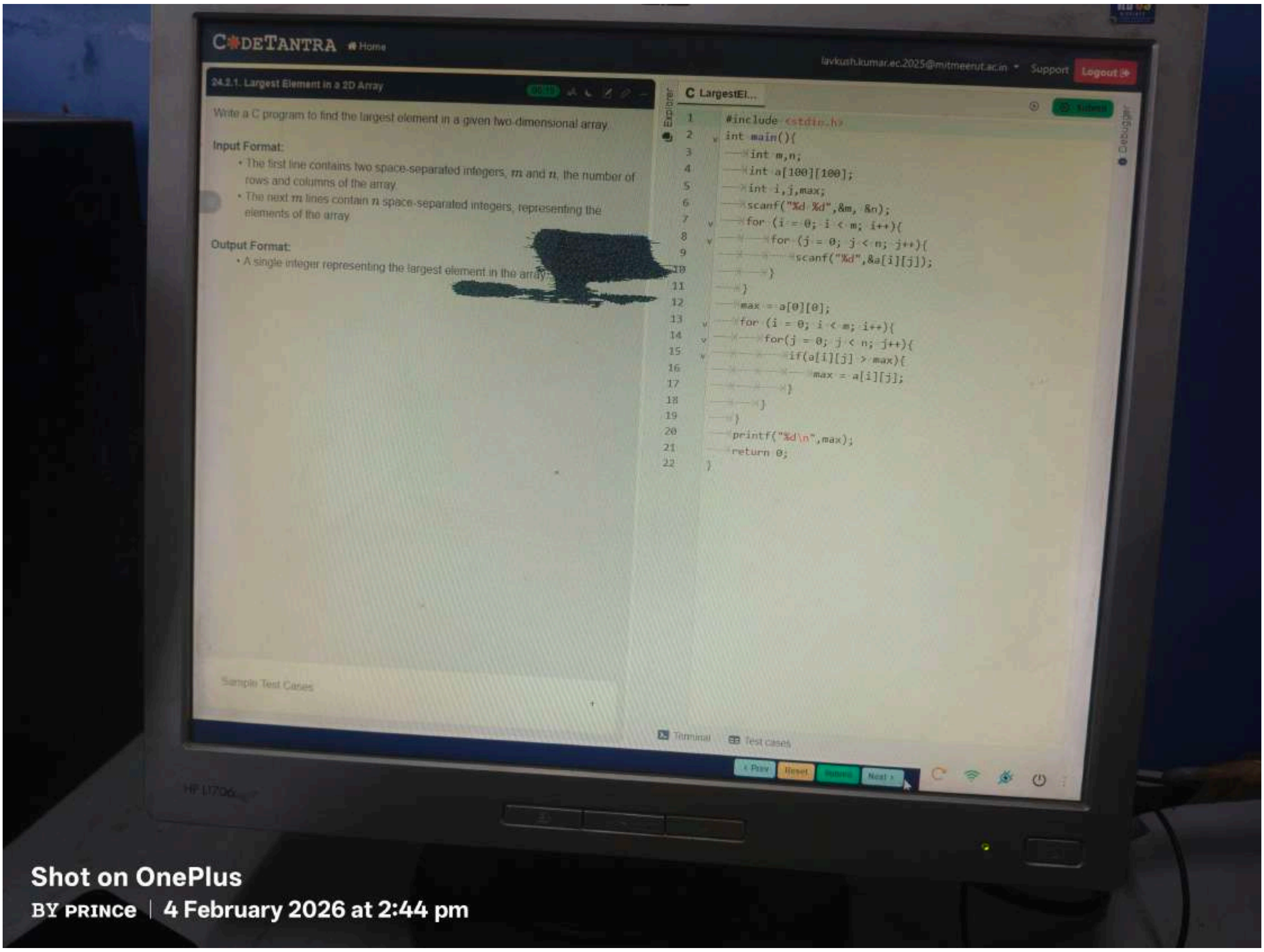
- 1 <= N <= 10^3
- 1 <= Elements of the array <= 10^6

Instruction: To run your custom test cases strictly map your input an [redacted] with the visible test cases

```c
#include <stdio.h>
int main (){
	int n, i;
	int a[100];
	int min, second_min;
	printf("No.of values: ");
	scanf("%d",&n);
	printf("Elements: ");
	for (i = 0; i < n; i++){
		scanf("%d",&a[i]);
	}
	if(a[0] < a[1]){
		min = a[0];
		second_min= a[1];
	}else{
		min = a[1];
		second_min= a[0];
	}
	for(i = 2; i < n; i++){
		if(a[i] < min){
			second_min= min;
			min = a[i];
		}else if (a[i] < second_min && a[i] != min){
			second_min = a[i];
		}
	}
	printf("Min element= %d\n",min);
	printf("Second min element= %d\n", second_min);
	return 0;
}
```

Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:44 pm





24.2.1. Largest Element in a 2D Array

Write a C program to find the largest element in a given two-dimensional array.

Input Format:

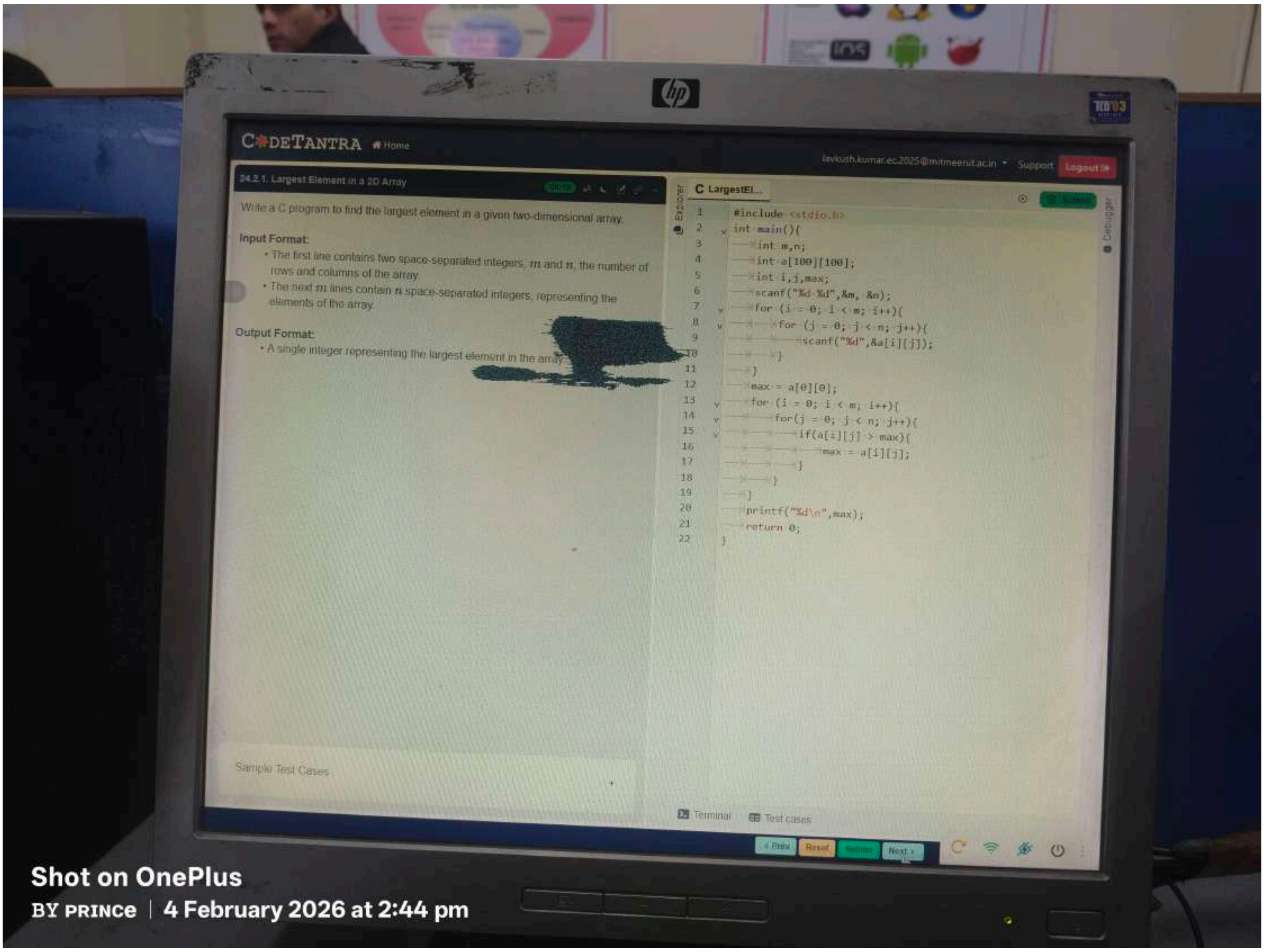
- The first line contains two space-separated integers, m and n, the number of rows and columns of the array.
- The next m lines contain n space-separated integers, representing the elements of the array.

Output Format:

- A single integer representing the largest element in the array. [redacted]

Sample Test Cases

```c
#include <stdio.h>
int main(){
    int m,n;
    int a[100][100];
    int i,j,max;
    scanf("%d %d",&m, &n);
    for (i = 0; i < m; i++){
        for (j = 0; j < n; j++){
            scanf("%d",&a[i][j]);
        }
    }
    max = a[0][0];
    for (i = 0; i < m; i++){
        for(j = 0; j < n; j++){
            if(a[i][j] > max){
                max = a[i][j];
            }
        }
    }
    printf("%d\n",max);
    return 0;
}
```



24.2.1. Largest Element in a 2D Array

Write a C program to find the largest element in a given two-dimensional array.

Input Format:

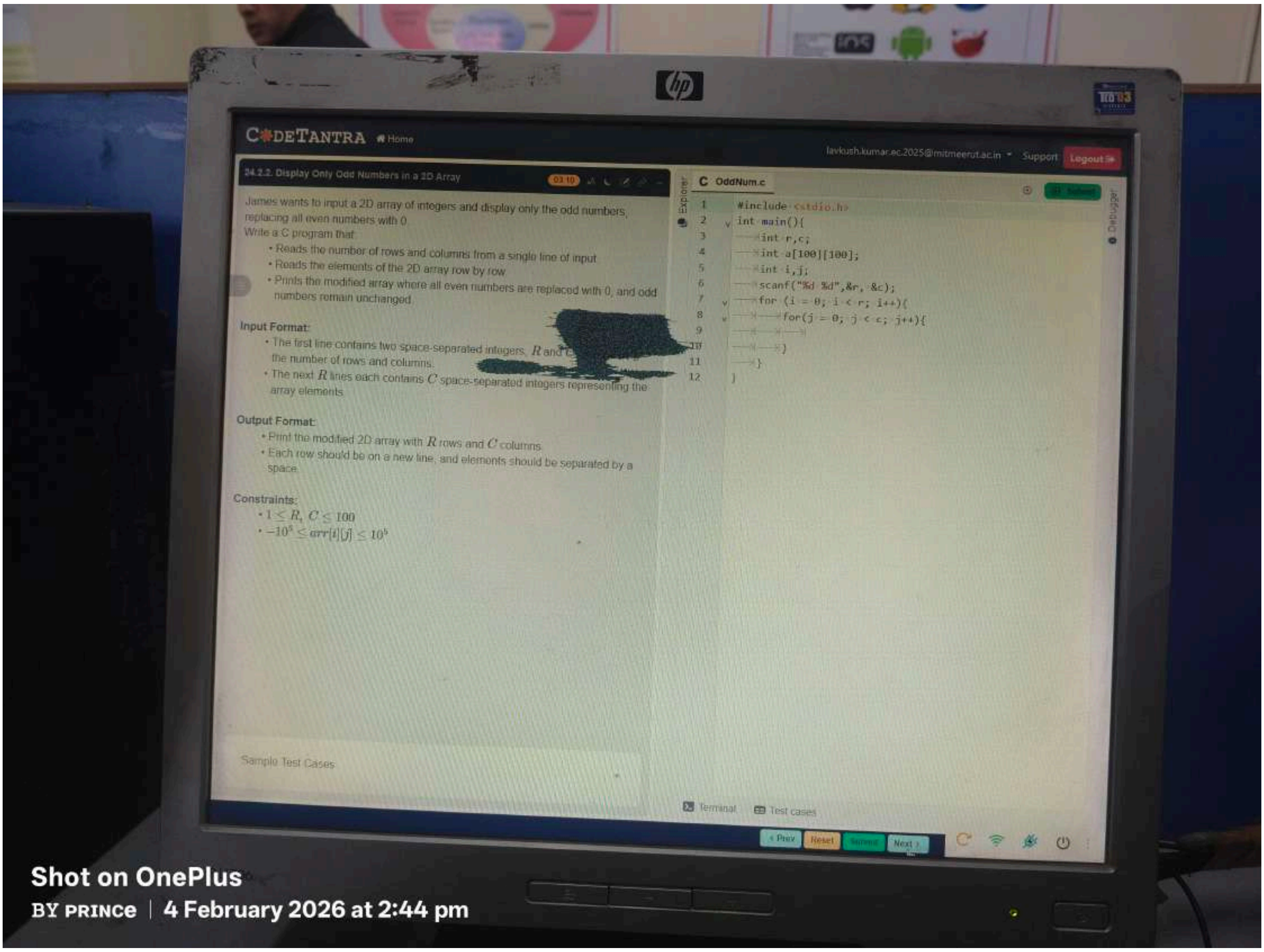
- The first line contains two space-separated integers, m and n, the number of rows and columns of the array.
- The next m lines contain n space-separated integers, representing the elements of the array.

Output Format:

- A single integer representing the largest element in the array.

[redacted]

```c
#include <stdio.h>
int main(){
    int m,n;
    int a[100][100];
    int i,j,max;
    scanf("%d %d",&m, &n);
    for (i = 0; i < m; i++){
        for (j = 0; j < n; j++){
            scanf("%d",&a[i][j]);
        }
    }
    max = a[0][0];
    for (i = 0; i < m; i++){
        for(j = 0; j < n; j++){
            if(a[i][j] > max){
                max = a[i][j];
            }
        }
    }
    printf("%d\n",max);
    return 0;
}
```



24.2.2. Display Only Odd Numbers in a 2D Array

James wants to input a 2D array of integers and display only the odd numbers, replacing all even numbers with 0

Write a C program that:

- Reads the number of rows and columns from a single line of input
- Reads the elements of the 2D array row by row
- Prints the modified array where all even numbers are replaced with 0, and odd numbers remain unchanged

Input Format:

- The first line contains two space-separated integers, R and [redacted] the number of rows and columns.
- The next R lines each contains C space-separated integers representing the array elements

Output Format:

- Print the modified 2D array with R rows and C columns
- Each row should be on a new line, and elements should be separated by a space.

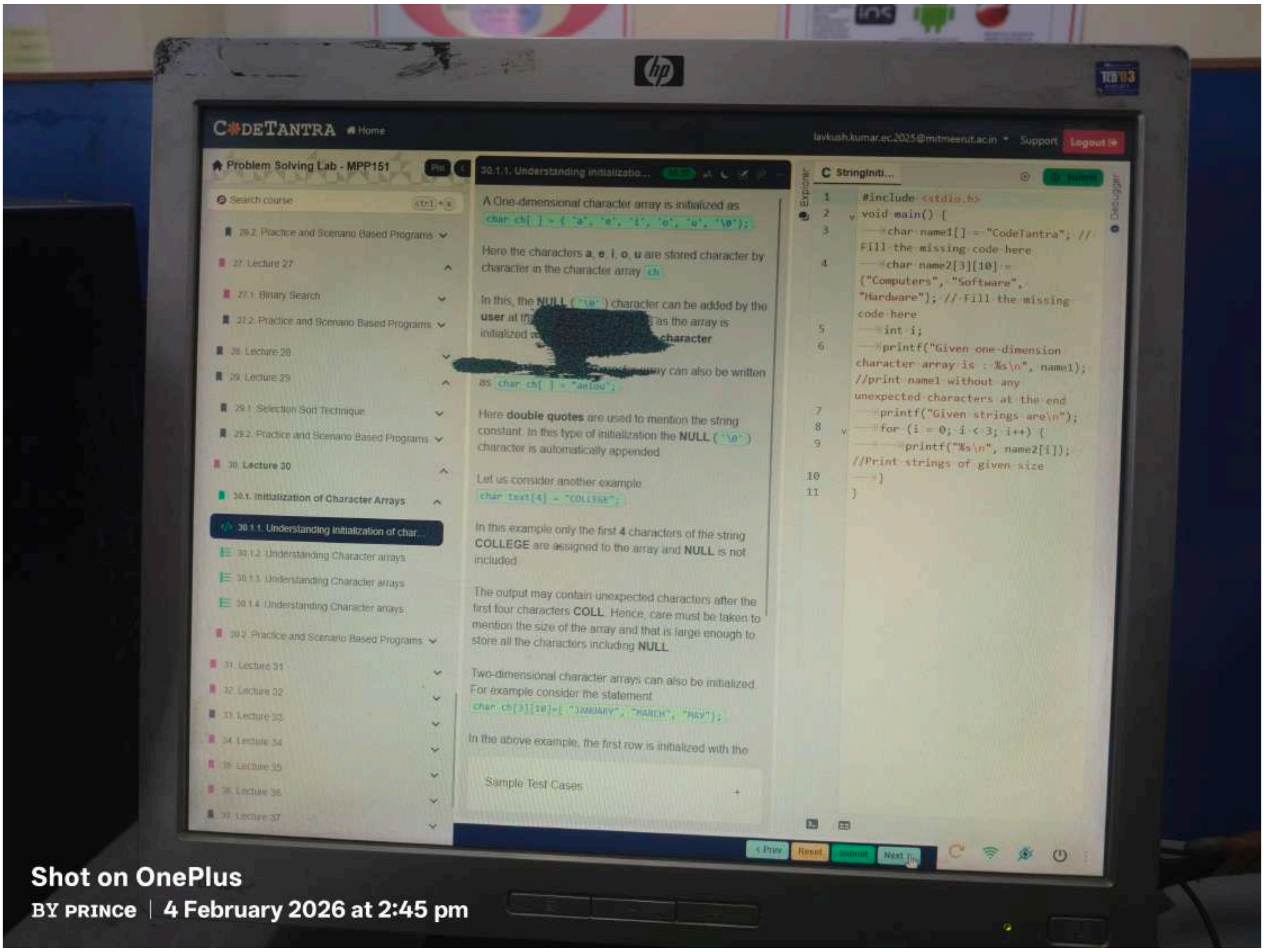
Constraints:

- $1 \leq R, C \leq 100$
- $-10^5 \leq arr[i][j] \leq 10^5$

Sample Test Cases

OddNum.c

```c
#include <stdio.h>
int main(){
	int r,c;
	int a[100][100];
	int i,j;
	scanf("%d %d",&r, &c);
	for (i = 0; i < r; i++){
		for(j = 0; j < c; j++){
			
		}
	}
}
```



C✱DETANTRA
Home
laykush.kumar.ec.2025@mitmeerut.ac.in
Support
Logout
Problem Solving Lab - MPP151
Search course
ctrl + k
26.2. Practice and Scenario Based Programs
27. Lecture 27
27.1. Binary Search
27.2. Practice and Scenario Based Programs
28. Lecture 28
29. Lecture 29
29.1. Selection Sort Technique
29.2. Practice and Scenario Based Programs
30. Lecture 30
30.1. Initialization of Character Arrays
30.1.1. Understanding initialization of char...
30.1.2. Understanding Character arrays
30.1.3. Understanding Character arrays
30.1.4. Understanding Character arrays
30.2. Practice and Scenario Based Programs
31. Lecture 31
32. Lecture 32
33. Lecture 33
34. Lecture 34
35. Lecture 35
36. Lecture 36
37. Lecture 37
30.1.1. Understanding initializatio...
Explorer
StringIniti...
Submit
Debugger
< Prev
Reset
Submit
Next

A One-dimensional character array is initialized as `char ch[ ] = { 'a', 'e', 'i', 'o', 'u', '\0'};`

Here the characters **a**, **e**, **i**, **o**, **u** are stored character by character in the character array `ch`

In this, the **NULL** (`'\0'`) character can be added by the **user** at th [redacted] as the array is initialized **character** ay can also be written as `char ch[ ] = "aeiou";`

Here **double quotes** are used to mention the string constant. In this type of initialization the **NULL** (`'\0'`) character is automatically appended

Let us consider another example `char text[4] = "COLLEGE";`

In this example only the first **4** characters of the string **COLLEGE** are assigned to the array and **NULL** is not included

The output may contain unexpected characters after the first four characters **COLL**. Hence, care must be taken to mention the size of the array and that is large enough to store all the characters including **NULL**

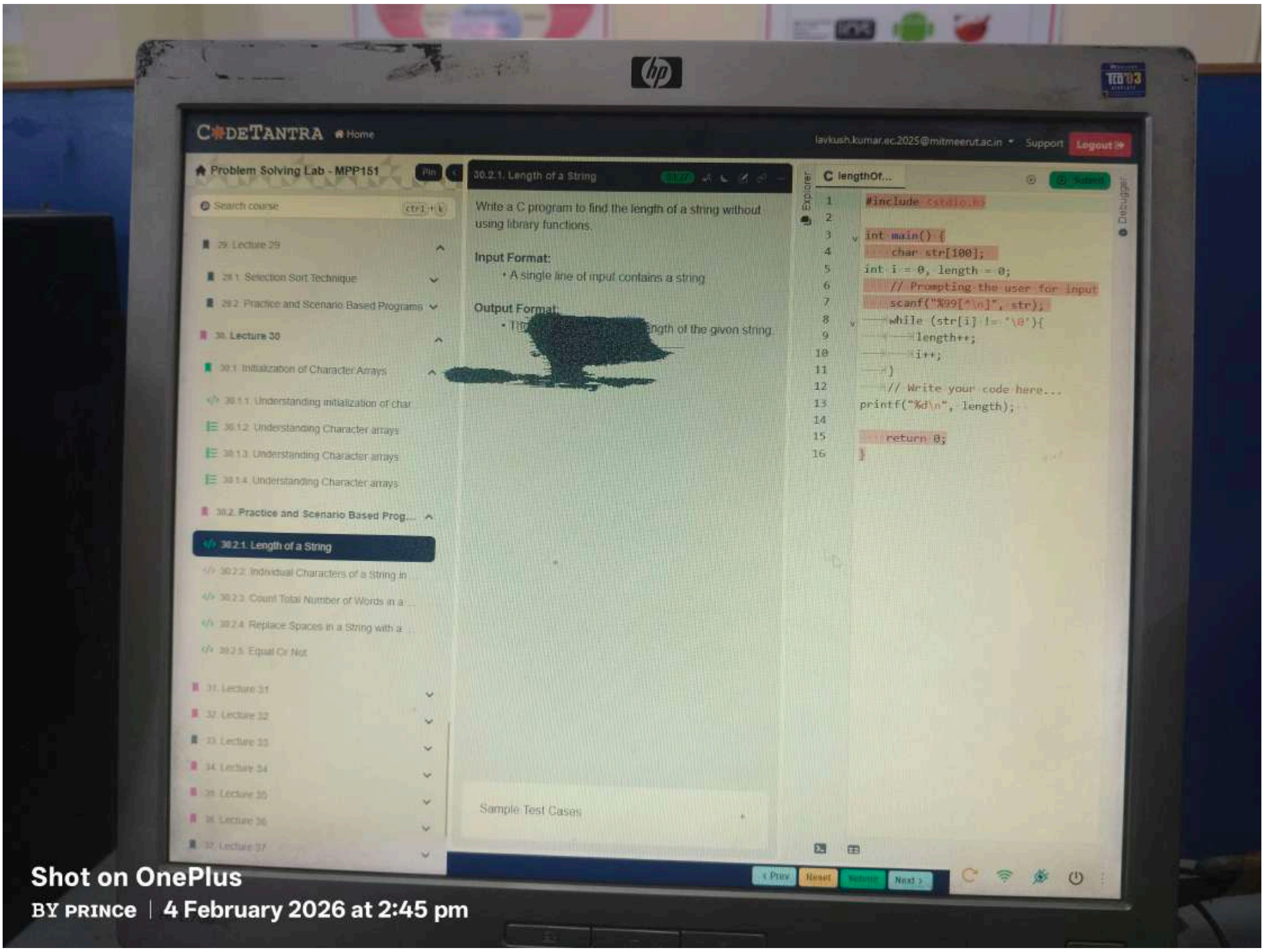
Two-dimensional character arrays can also be initialized. For example consider the statement `char ch[3][10]={ "JANUARY", "MARCH", "MAY"};`

In the above example, the first row is initialized with the

Sample Test Cases

```c
#include <stdio.h>
void main() {
	char name1[] = "CodeTantra"; // Fill the missing code here
	char name2[3][10] = {"Computers", "Software", "Hardware"}; // Fill the missing code here
	int i;
	printf("Given one-dimension character array is : %s\n", name1); //print name1 without any unexpected characters at the end
	printf("Given strings are\n");
	for (i = 0; i < 3; i++) {
		printf("%s\n", name2[i]); //Print strings of given size
	}
}
```

Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:45 pm



C✱DETANTRA Home

laykush.kumar.ec.2025@mitmeerut.ac.in Support Logout

Problem Solving Lab - MPP151

Search course ctrl + k

- 29. Lecture 29
  - 29.1. Selection Sort Technique
  - 29.2. Practice and Scenario Based Programs
- 30. Lecture 30
  - 30.1. Initialization of Character Arrays
    - 30.1.1. Understanding initialization of char
    - 30.1.2. Understanding Character arrays
    - 30.1.3. Understanding Character arrays
    - 30.1.4. Understanding Character arrays
  - 30.2. Practice and Scenario Based Prog...
    - 30.2.1. Length of a String
    - 30.2.2. Individual Characters of a String in
    - 30.2.3. Count Total Number of Words in a ...
    - 30.2.4. Replace Spaces in a String with a ...
    - 30.2.5. Equal Or Not
- 31. Lecture 31
- 32. Lecture 32
- 33. Lecture 33
- 34. Lecture 34
- 35. Lecture 35
- 36. Lecture 36
- 37. Lecture 37

30.2.1. Length of a String

Write a C program to find the length of a string without using library functions.

Input Format:

- A single line of input contains a string

Output Format:

- Th[redacted]ngth of the given string.

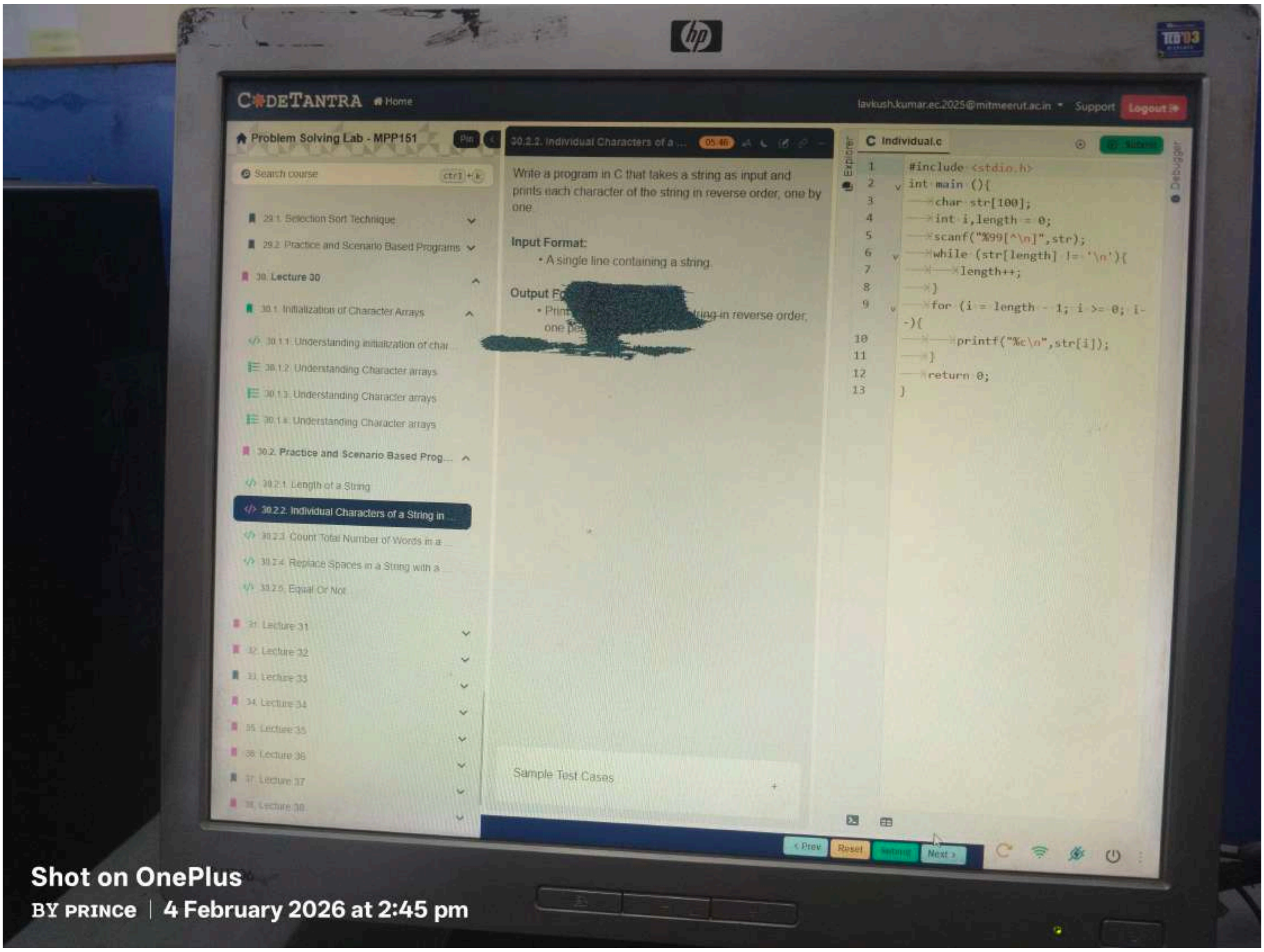
Sample Test Cases

lengthOf...

```c
#include <stdio.h>

int main() {
    char str[100];
int i = 0, length = 0;
    // Prompting the user for input
    scanf("%99[^\n]", str);
    while (str[i] != '\0'){
        length++;
        i++;
    }
    // Write your code here...
printf("%d\n", length);

    return 0;
}
```

Shot on OnePlus

BY PRINCE | 4 February 2026 at 2:45 pm


CODETANTRA
Home
lavkush.kumar.ec.2025@mitmeerut.ac.in
Support
Logout
Problem Solving Lab - MPP151
Pin
Search course
ctrl + k
29.1. Selection Sort Technique
29.2. Practice and Scenario Based Programs
30. Lecture 30
30.1. Initialization of Character Arrays
30.1.1. Understanding initialization of char...
30.1.2. Understanding Character arrays
30.1.3. Understanding Character arrays
30.1.4. Understanding Character arrays
30.2. Practice and Scenario Based Prog...
30.2.1. Length of a String
30.2.2. Individual Characters of a String in ...
30.2.3. Count Total Number of Words in a ...
30.2.4. Replace Spaces in a String with a ...
30.2.5. Equal Or Not
31. Lecture 31
32. Lecture 32
33. Lecture 33
34. Lecture 34
35. Lecture 35
36. Lecture 36
37. Lecture 37
38. Lecture 38
30.2.2. Individual Characters of a ...
05:46
Explorer
Individual.c
Submit
Debugger
Sample Test Cases
< Prev
Reset
Submit
Next >
Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:45 pm

Write a program in C that takes a string as input and prints each character of the string in reverse order, one by one.

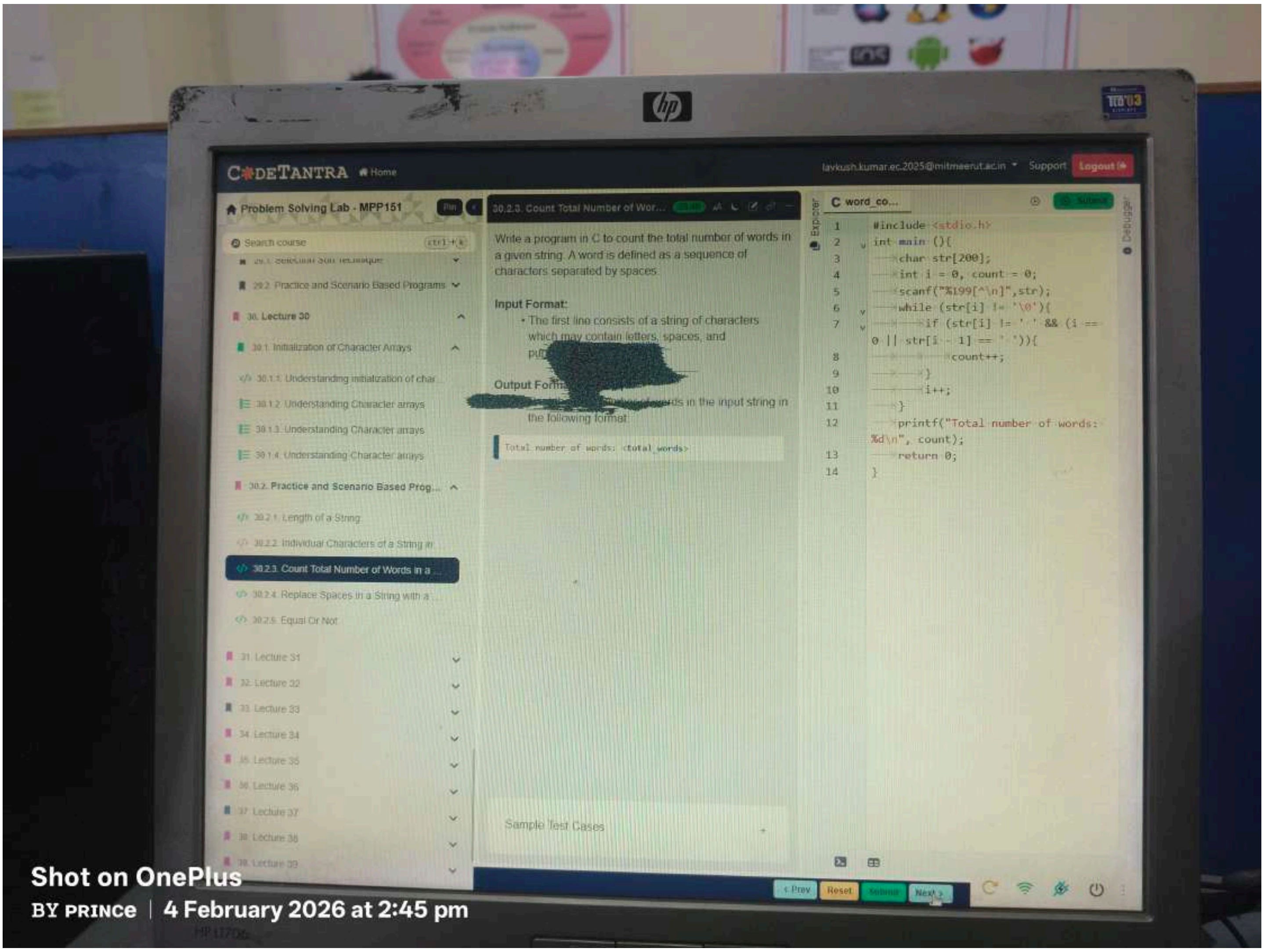
Input Format:

- A single line containing a string.

Output Fo[redacted]

- Prin[redacted]tring in reverse order, one pe[redacted]

```c
#include <stdin.h>
int main (){
    char str[100];
    int i,length = 0;
    scanf("%99[^\n]",str);
    while (str[length] != '\n'){
        length++;
    }
    for (i = length - 1; i >= 0; i--){
        printf("%c\n",str[i]);
    }
    return 0;
}
```


CODETANTRA
Home
laykush.kumar.ec.2025@mitmeerut.ac.in
Support
Logout
Problem Solving Lab - MPP151
Pin
Search course
ctrl + k
29.1. Selection Sort Technique
29.2. Practice and Scenario Based Programs
30. Lecture 30
30.1. Initialization of Character Arrays
30.1.1. Understanding initialization of char...
30.1.2. Understanding Character arrays
30.1.3. Understanding Character arrays
30.1.4. Understanding Character arrays
30.2. Practice and Scenario Based Prog...
30.2.1. Length of a String
30.2.2. Individual Characters of a String in
30.2.3. Count Total Number of Words in a ...
30.2.4. Replace Spaces in a String with a ...
30.2.5. Equal Or Not
31. Lecture 31
32. Lecture 32
33. Lecture 33
34. Lecture 34
35. Lecture 35
36. Lecture 36
37. Lecture 37
38. Lecture 38
39. Lecture 39
30.2.3. Count Total Number of Wor...
Explorer
C word_co...
Submit
Debugger
Sample Test Cases
< Prev
Reset
Submit
Next >
Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:45 pm

Write a program in C to count the total number of words in a given string. A word is defined as a sequence of characters separated by spaces.

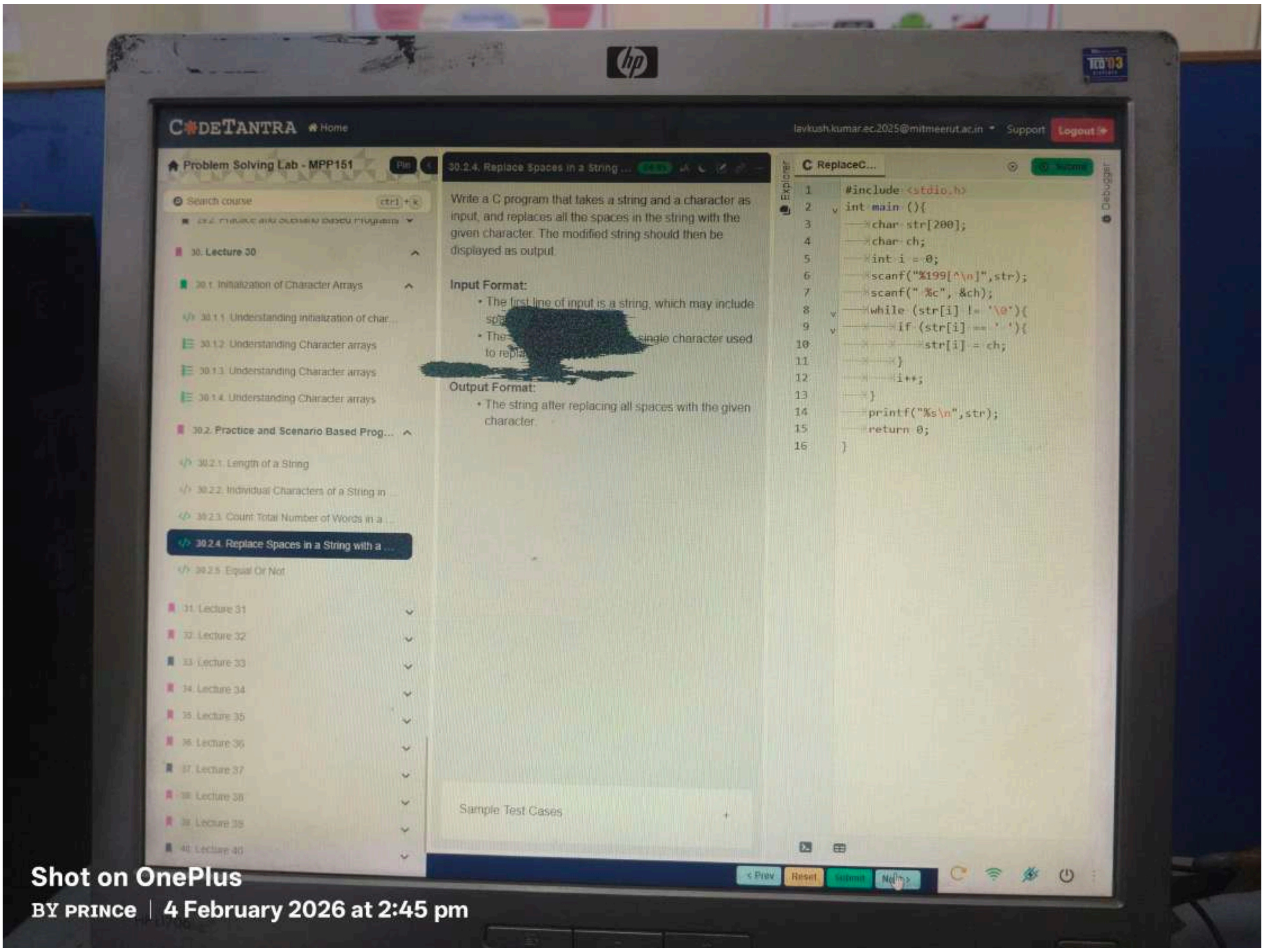
Input Format:

- The first line consists of a string of characters which may contain letters, spaces, and pu[redacted]

Output Form[redacted]ds in the input string in the following format:

```
Total number of words: <total_words>
```

```c
#include <stdio.h>
int main (){
	char str[200];
	int i = 0, count = 0;
	scanf("%199[^\n]",str);
	while (str[i] != '\0'){
		if (str[i] != ' ' && (i == 0 || str[i - 1] == ' ')){
			count++;
		}
		i++;
	}
	printf("Total number of words: %d\n", count);
	return 0;
}
```



CODETANTRA
Home
lavkush.kumar.ec.2025@mitmeerut.ac.in
Support
Logout
Problem Solving Lab - MPP151
Search course
ctrl + k
30. Lecture 30
30.1. Initialization of Character Arrays
30.1.1. Understanding initialization of char...
30.1.2. Understanding Character arrays
30.1.3. Understanding Character arrays
30.1.4. Understanding Character arrays
30.2. Practice and Scenario Based Prog...
30.2.1. Length of a String
30.2.2. Individual Characters of a String in ...
30.2.3. Count Total Number of Words in a ...
30.2.4. Replace Spaces in a String with a ...
30.2.5. Equal Or Not
31. Lecture 31
32. Lecture 32
33. Lecture 33
34. Lecture 34
35. Lecture 35
36. Lecture 36
37. Lecture 37
38. Lecture 38
39. Lecture 39
40. Lecture 40
30.2.4. Replace Spaces in a String ...
Explorer
C ReplaceC...
Debugger
Sample Test Cases
< Prev
Reset
Submit
Next >

Write a C program that takes a string and a character as input, and replaces all the spaces in the string with the given character. The modified string should then be displayed as output.

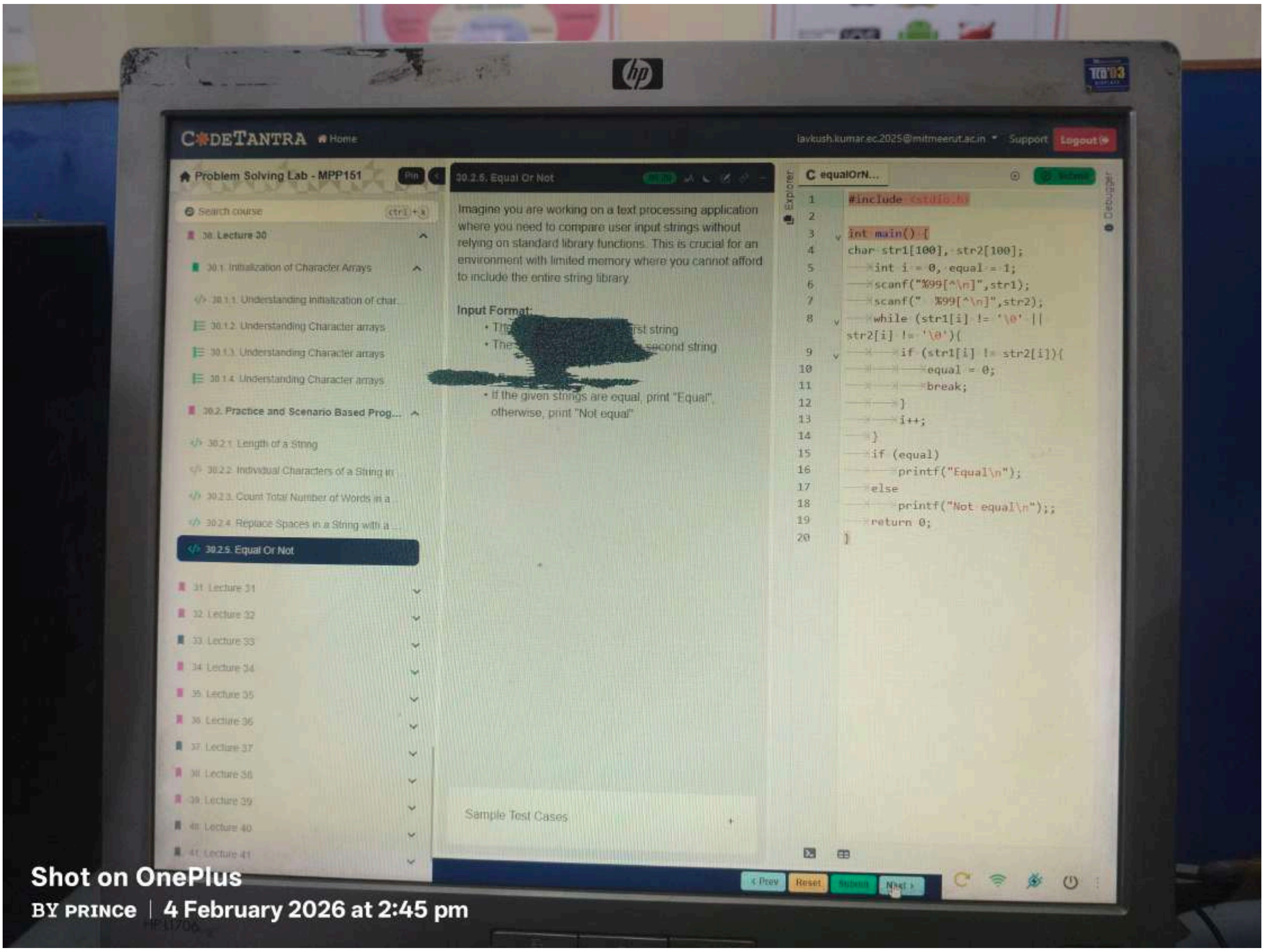
Input Format:

- The first line of input is a string, which may include [redacted] single character used

Output Format:

- The string after replacing all spaces with the given character.

```c
#include <stdio.h>
int main (){
	char str[200];
	char ch;
	int i = 0;
	scanf("%199[^\n]",str);
	scanf(" %c", &ch);
	while (str[i] != '\0'){
		if (str[i] == ' '){
			str[i] = ch;
		}
		i++;
	}
	printf("%s\n",str);
	return 0;
}
```


CODETANTRA
Home
lavkush.kumar.ec.2025@mitmeerut.ac.in
Support
Logout
Problem Solving Lab - MPP151
Pin
Search course
ctrl + k
30. Lecture 30
30.1. Initialization of Character Arrays
30.1.1. Understanding initialization of char...
30.1.2. Understanding Character arrays
30.1.3. Understanding Character arrays
30.1.4. Understanding Character arrays
30.2. Practice and Scenario Based Prog...
30.2.1. Length of a String
30.2.2. Individual Characters of a String in ...
30.2.3. Count Total Number of Words in a ...
30.2.4. Replace Spaces in a String with a ...
30.2.5. Equal Or Not
31. Lecture 31
32. Lecture 32
33. Lecture 33
34. Lecture 34
35. Lecture 35
36. Lecture 36
37. Lecture 37
38. Lecture 38
39. Lecture 39
40. Lecture 40
41. Lecture 41
Explorer
Debugger
C equalOrN...
Submit
< Prev
Reset
Submit
Next >
Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:45 pm

30.2.5. Equal Or Not

Imagine you are working on a text processing application where you need to compare user input strings without relying on standard library functions. This is crucial for an environment with limited memory where you cannot afford to include the entire string library.

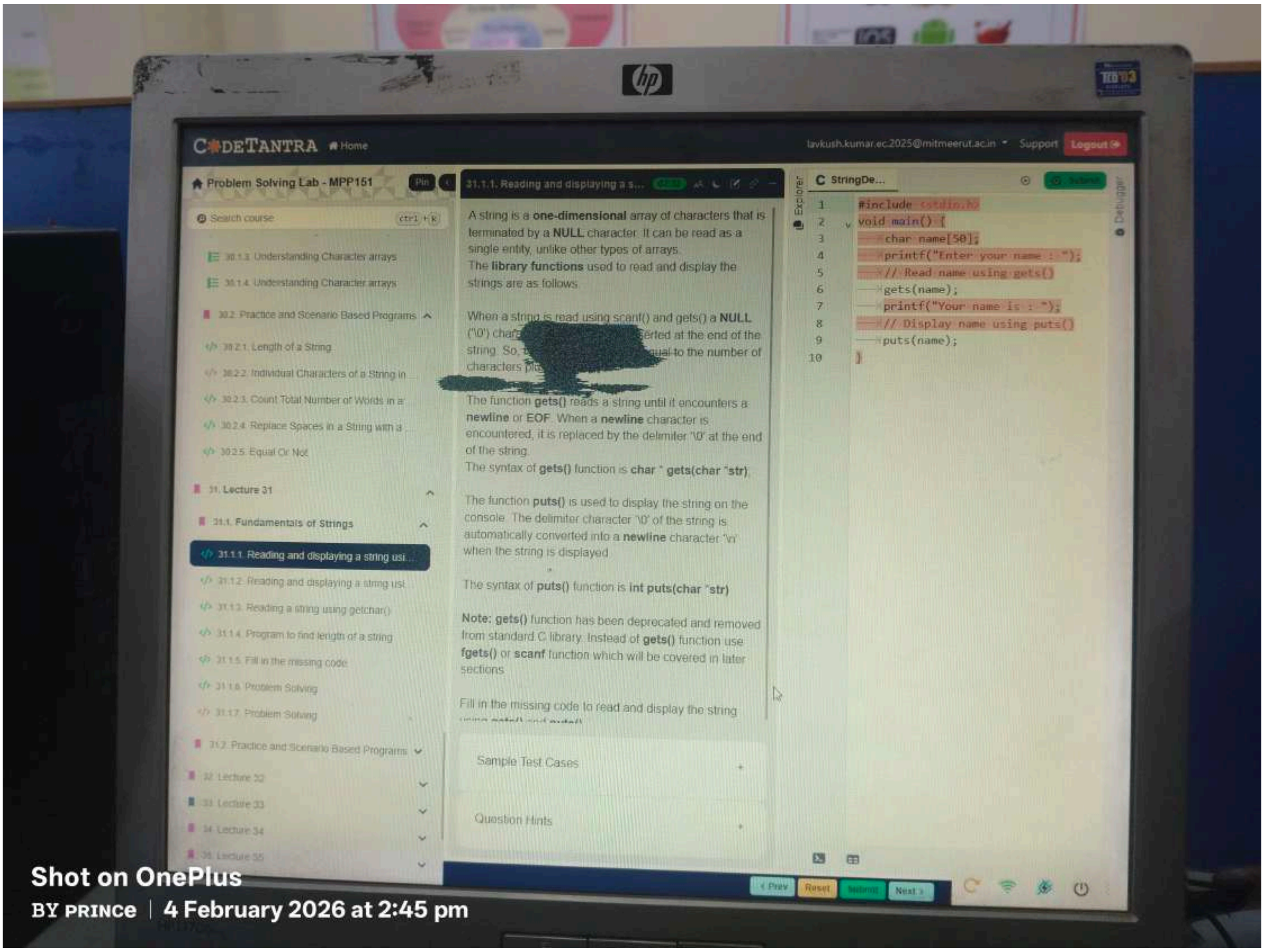
Input Format:

- Th [redacted] rst string
- The second string

- If the given strings are equal, print "Equal", otherwise, print "Not equal"

Sample Test Cases

```c
#include <stdio.h>

int main() {
char str1[100], str2[100];
    int i = 0, equal = 1;
    scanf("%99[^\n]",str1);
    scanf("  %99[^\n]",str2);
    while (str1[i] != '\0' ||
str2[i] != '\0'){
        if (str1[i] != str2[i]){
            equal = 0;
            break;
        }
        i++;
    }
    if (equal)
        printf("Equal\n");
    else
        printf("Not equal\n");;
    return 0;
}
```


C*DETANTRA
Home
lavkush.kumar.ec2025@mitmeerut.ac.in
Support
Logout
Problem Solving Lab - MPP151
Pin
Search course
ctrl + k
30.1.3. Understanding Character arrays
30.1.4. Understanding Character arrays
30.2. Practice and Scenario Based Programs
30.2.1. Length of a String
30.2.2. Individual Characters of a String in ...
30.2.3. Count Total Number of Words in a ...
30.2.4. Replace Spaces in a String with a ...
30.2.5. Equal Or Not
31. Lecture 31
31.1. Fundamentals of Strings
31.1.1. Reading and displaying a string usi...
31.1.2. Reading and displaying a string usi...
31.1.3. Reading a string using getchar()
31.1.4. Program to find length of a string
31.1.5. Fill in the missing code
31.1.6. Problem Solving
31.1.7. Problem Solving
31.2. Practice and Scenario Based Programs
32. Lecture 32
33. Lecture 33
34. Lecture 34
35. Lecture 35
31.1.1. Reading and displaying a s...
Explorer
StringDe...
Submit
Debugger
Sample Test Cases
Question Hints
< Prev
Reset
Submit
Next >

A string is a **one-dimensional** array of characters that is terminated by a **NULL** character. It can be read as a single entity, unlike other types of arrays.
The **library functions** used to read and display the strings are as follows.

When a string is read using scanf() and gets() a **NULL** ('\0') chara[redacted]erted at the end of the string. So, t ... qual to the number of characters plu

The function **gets()** reads a string until it encounters a **newline** or EOF. When a **newline** character is encountered, it is replaced by the delimiter '\0' at the end of the string.
The syntax of **gets()** function is **char * gets(char *str)**.

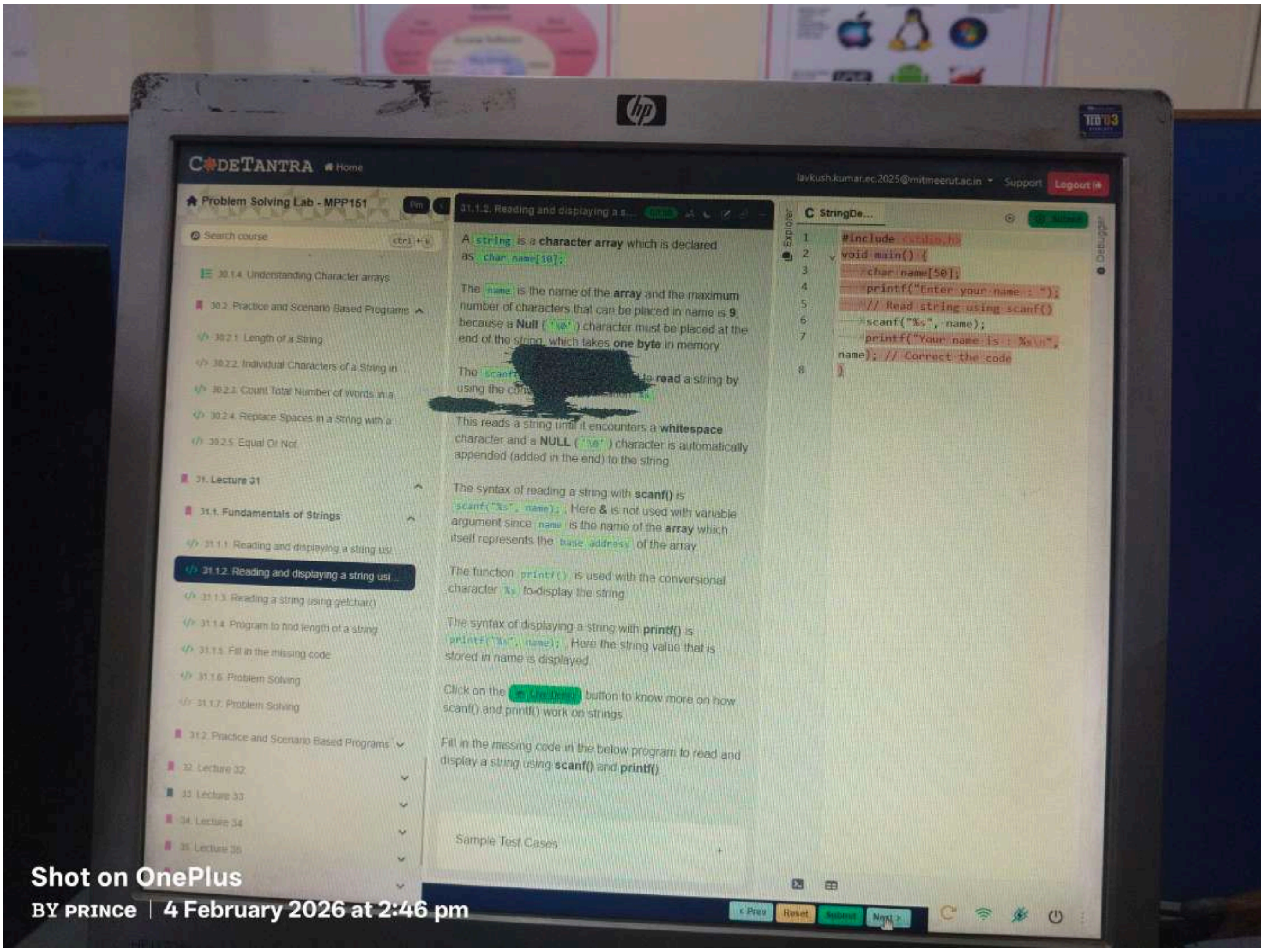
The function **puts()** is used to display the string on the console. The delimiter character '\0' of the string is automatically converted into a **newline** character '\n' when the string is displayed.

The syntax of **puts()** function is **int puts(char *str)**

Note: gets() function has been deprecated and removed from standard C library. Instead of **gets()** function use **fgets()** or **scanf** function which will be covered in later sections.

Fill in the missing code to read and display the string

```c
#include <stdio.h>
void main() {
	char name[50];
	printf("Enter your name : ");
	// Read name using gets()
	gets(name);
	printf("Your name is : ");
	// Display name using puts()
	puts(name);
}
```


CODETANTRA
Home
lavkush.kumar.ec.2025@mitmeerut.ac.in
Support
Logout
Problem Solving Lab - MPP151
Search course
ctrl + k
30.1.4. Understanding Character arrays
30.2. Practice and Scenario Based Programs
30.2.1. Length of a String
30.2.2. Individual Characters of a String in
30.2.3. Count Total Number of Words in a
30.2.4. Replace Spaces in a String with a
30.2.5. Equal Or Not
31. Lecture 31
31.1. Fundamentals of Strings
31.1.1. Reading and displaying a string usi
31.1.2. Reading and displaying a string usi...
31.1.3. Reading a string using getchar()
31.1.4. Program to find length of a string
31.1.5. Fill in the missing code
31.1.6. Problem Solving
31.1.7. Problem Solving
31.2. Practice and Scenario Based Programs
32. Lecture 32
33. Lecture 33
34. Lecture 34
35. Lecture 35
31.1.2. Reading and displaying a s...
Explorer
C StringDe...
Debugger
Sample Test Cases
< Prev
Reset
Submit
Next >
Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:46 pm

A `string` is a **character array** which is declared as `char name[10];`.

The `name` is the name of the **array** and the maximum number of characters that can be placed in name is **9**, because a **Null** (`'\0'`) character must be placed at the end of the string, which takes **one byte** in memory.

The `scanf` [redacted] to **read** a string by using the con[redacted]

This reads a string until it encounters a **whitespace** character and a **NULL** (`'\0'`) character is automatically appended (added in the end) to the string.

The syntax of reading a string with **scanf()** is `scanf("%s", name);`. Here **&** is not used with variable argument since `name` is the name of the **array** which itself represents the `base address` of the array.

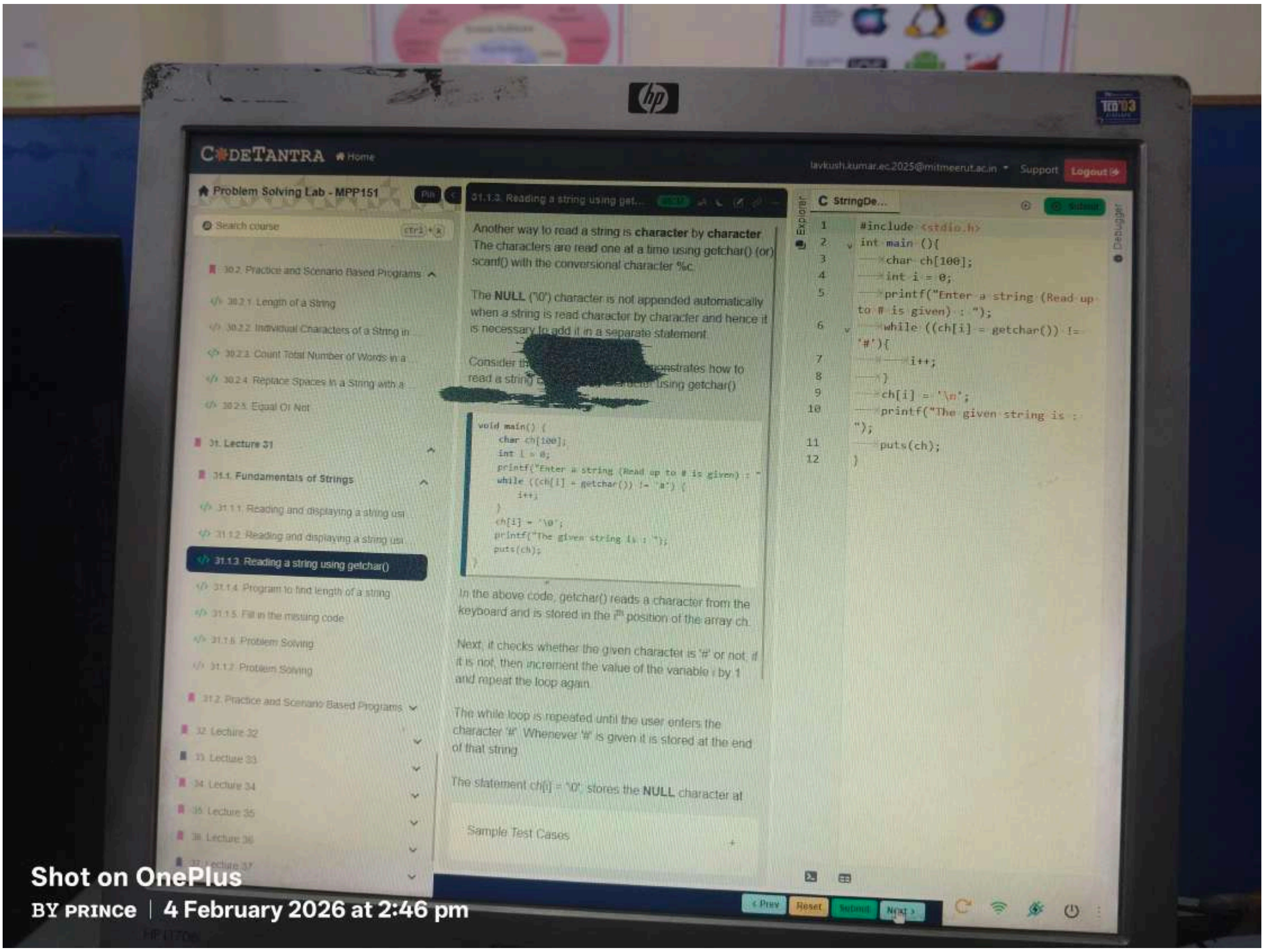
The function `printf()` is used with the conversional character `%s` to-display the string.

The syntax of displaying a string with **printf()** is `printf("%s", name);`. Here the string value that is stored in name is displayed.

Click on the Live Demo button to know more on how scanf() and printf() work on strings.

Fill in the missing code in the below program to read and display a string using **scanf()** and **printf()**.

```c
#include <stdio.h>
void main() {
	char name[50];
	printf("Enter your name : ");
	// Read string using scanf()
	scanf("%s", name);
	printf("Your name is : %s\n", 
name); // Correct the code
}
```


CODETANTRA
Home
lavkush.kumar.ec2025@mitmeerut.ac.in
Support
Logout
Problem Solving Lab - MPP151
Pin
Search course
ctrl + k
30.2. Practice and Scenario Based Programs
30.2.1. Length of a String
30.2.2. Individual Characters of a String in ...
30.2.3. Count Total Number of Words in a ...
30.2.4. Replace Spaces in a String with a ...
30.2.5. Equal Or Not
31. Lecture 31
31.1. Fundamentals of Strings
31.1.1. Reading and displaying a string usi...
31.1.2. Reading and displaying a string usi...
31.1.3. Reading a string using getchar()
31.1.4. Program to find length of a string
31.1.5. Fill in the missing code
31.1.6. Problem Solving
31.1.7. Problem Solving
31.2. Practice and Scenario Based Programs
32. Lecture 32
33. Lecture 33
34. Lecture 34
35. Lecture 35
36. Lecture 36
37. Lecture 37
31.1.3. Reading a string using get...
Explorer
StringDe...
Submit
Debugger
Sample Test Cases
< Prev
Reset
Submit
Next >
Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:46 pm

Another way to read a string is **character** by **character**. The characters are read one at a time using getchar() (or) scanf() with the conversional character %c.

The **NULL** ('\0') character is not appended automatically when a string is read character by character and hence it is necessary to add it in a separate statement.

Consider th [redacted] nstrates how to read a string c [redacted] using getchar()

```
void main() {
    char ch[100];
    int i = 0;
    printf("Enter a string (Read up to # is given) : -
    while ((ch[i] = getchar()) != '#') {
        i++;
    }
    ch[i] = '\0';
    printf("The given string is : ");
    puts(ch);
}
```

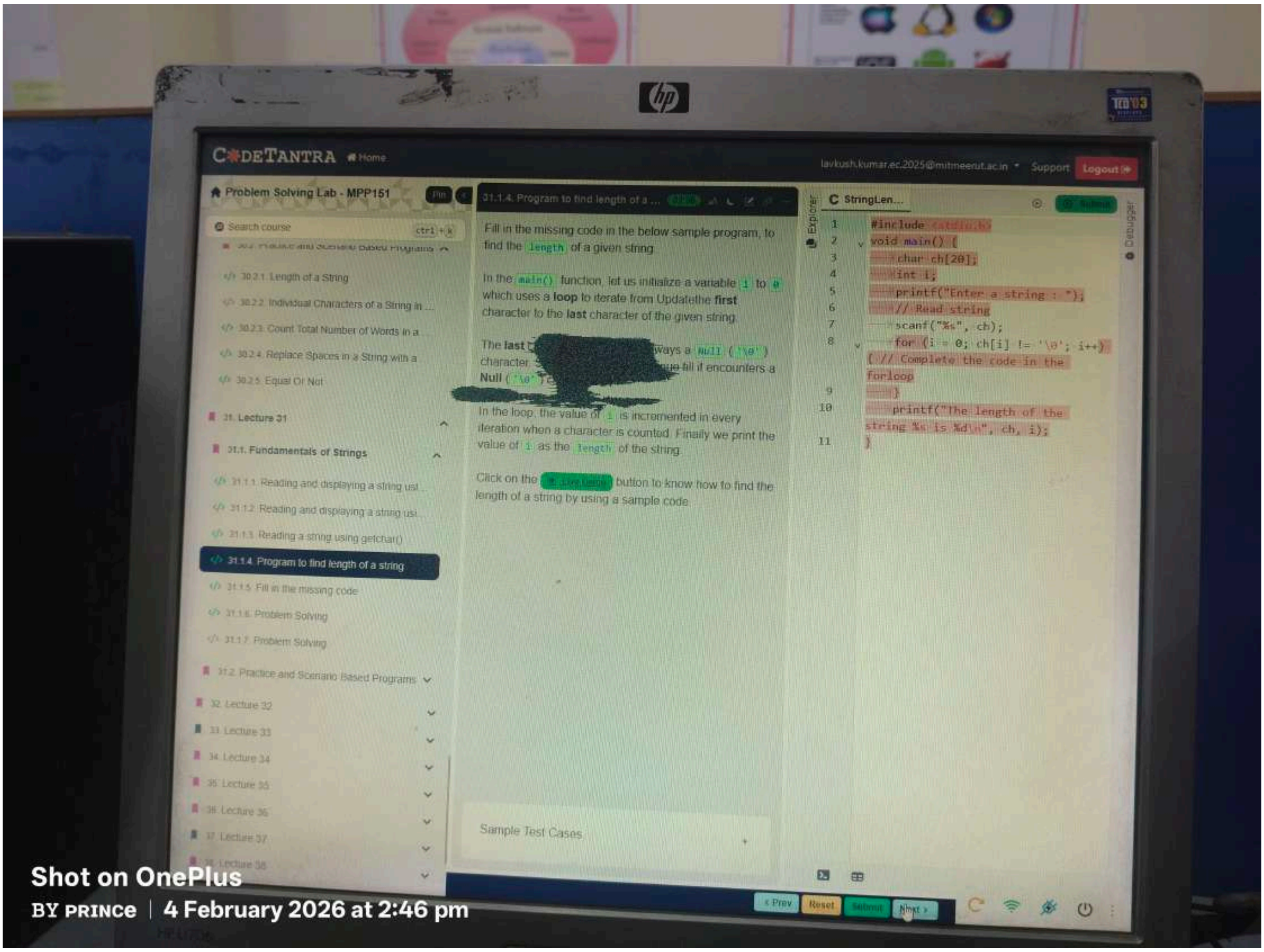
In the above code, getchar() reads a character from the keyboard and is stored in the i^{th} position of the array ch.

Next, it checks whether the given character is '#' or not, if it is not, then increment the value of the variable i by 1 and repeat the loop again.

The while loop is repeated until the user enters the character '#'. Whenever '#' is given it is stored at the end of that string.

The statement ch[i] = '\0', stores the **NULL** character at

```
#include <stdio.h>
int main (){
    char ch[100];
    int i = 0;
    printf("Enter a string (Read up to # is given) : ");
    while ((ch[i] = getchar()) != '#'){
        i++;
    }
    ch[i] = '\n';
    printf("The given string is : ");
    puts(ch);
}
```



CODETANTRA
Home
lavkush.kumar.ec.2025@mitmeerut.ac.in
Support
Logout
Problem Solving Lab - MPP151
Pin
Search course
ctrl + k
30.2. Practice and Scenario Based Programs
30.2.1. Length of a String
30.2.2. Individual Characters of a String in ...
30.2.3. Count Total Number of Words in a ...
30.2.4. Replace Spaces in a String with a ...
30.2.5. Equal Or Not
31. Lecture 31
31.1. Fundamentals of Strings
31.1.1. Reading and displaying a string usi...
31.1.2. Reading and displaying a string usi...
31.1.3. Reading a string using getchar()
31.1.4. Program to find length of a string
31.1.5. Fill in the missing code
31.1.6. Problem Solving
31.1.7. Problem Solving
31.2. Practice and Scenario Based Programs
32. Lecture 32
33. Lecture 33
34. Lecture 34
35. Lecture 35
36. Lecture 36
37. Lecture 37
38. Lecture 38
31.1.4. Program to find length of a ...
Explorer
C StringLen...
Submit
Debugger
Sample Test Cases
< Prev
Reset
Submit
Next >

Fill in the missing code in the below sample program, to find the `length` of a given string.

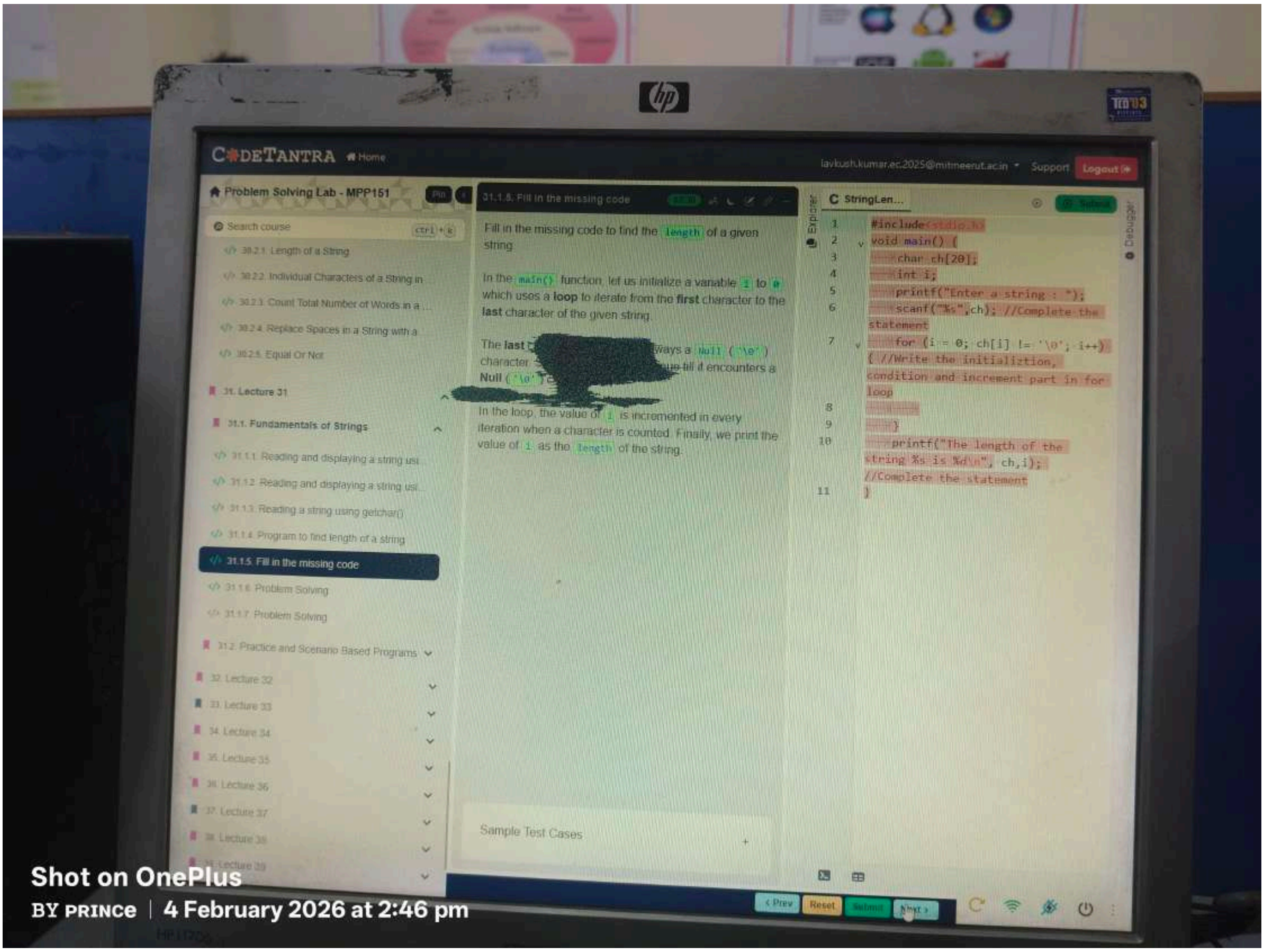
In the `main()` function, let us initialize a variable `i` to `0` which uses a **loop** to iterate from Updatethe **first** character to the **last** character of the given string.

The **last** [redacted] ways a `Null` (`'\0'`) character [redacted] ue till it encounters a **Null** (`'\0'`) [redacted]

In the loop, the value of `i` is incremented in every iteration when a character is counted. Finally we print the value of `i` as the `length` of the string.

Click on the Live Demo button to know how to find the length of a string by using a sample code.

```c
#include <stdio.h>
void main() {
	char ch[20];
	int i;
	printf("Enter a string : ");
	// Read string
	scanf("%s", ch);
	for (i = 0; ch[i] != '\0'; i++)
{ // Complete the code in the forloop
	}
	printf("The length of the string %s is %d\n", ch, i);
}
```


CODETANTRA
Home
Problem Solving Lab - MPP151
Search course
30.2.1. Length of a String
30.2.2. Individual Characters of a String in
30.2.3. Count Total Number of Words in a ...
30.2.4. Replace Spaces in a String with a
30.2.5. Equal Or Not
31. Lecture 31
31.1. Fundamentals of Strings
31.1.1. Reading and displaying a string usi...
31.1.2. Reading and displaying a string usi...
31.1.3. Reading a string using getchar()
31.1.4. Program to find length of a string
31.1.5. Fill in the missing code
31.1.6. Problem Solving
31.1.7. Problem Solving
31.2. Practice and Scenario Based Programs
32. Lecture 32
33. Lecture 33
34. Lecture 34
35. Lecture 35
36. Lecture 36
37. Lecture 37
38. Lecture 38
39. Lecture 39
lavkush.kumar.ec.2025@mitmeerut.ac.in
Support
Logout
Explorer
StringLen...
Submit
Debugger
Sample Test Cases
< Prev
Reset
Submit
Next >

31.1.5. Fill in the missing code

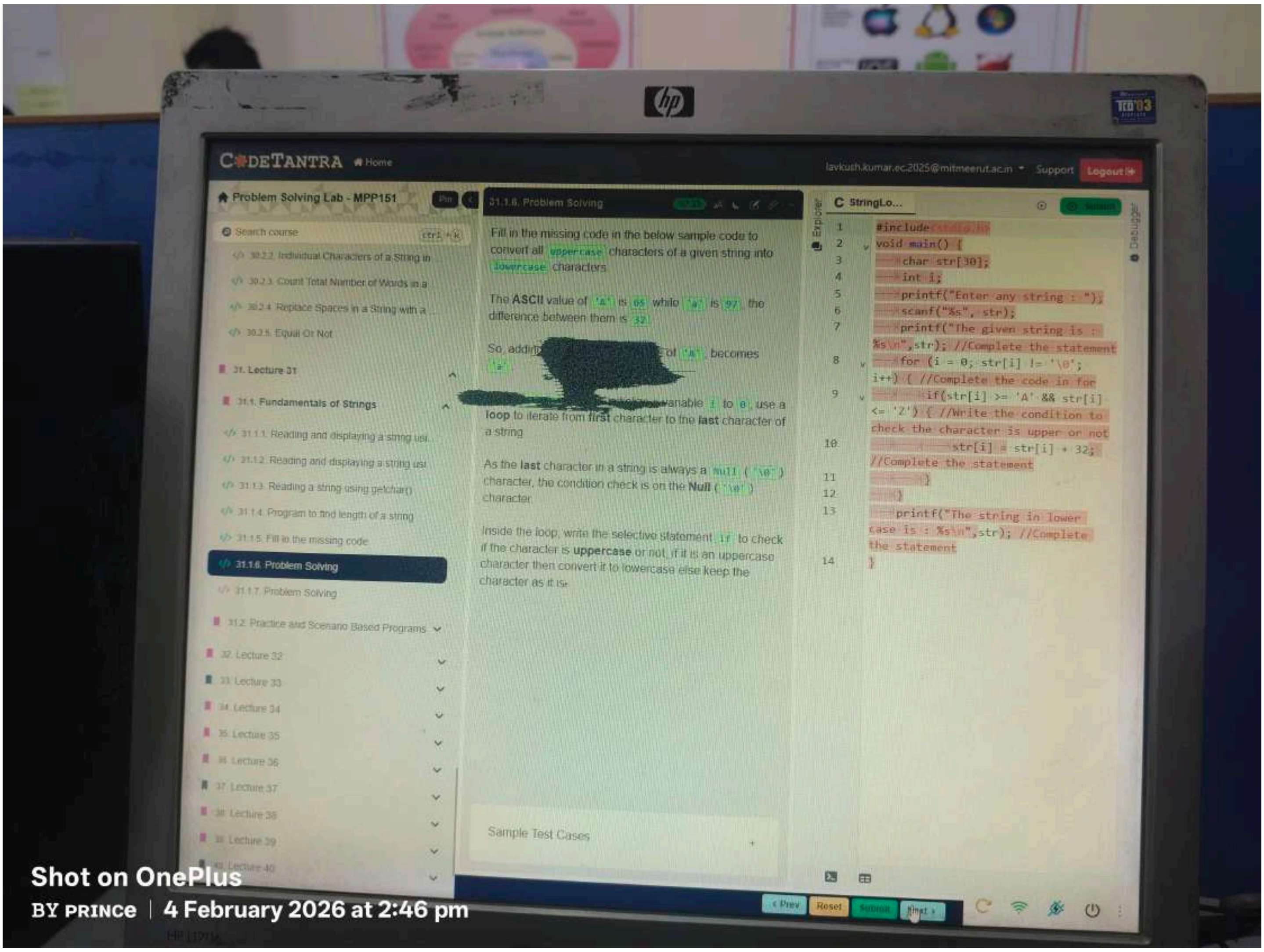
Fill in the missing code to find the `length` of a given string.

In the `main()` function, let us initialize a variable `i` to `0` which uses a **loop** to iterate from the **first** character to the **last** character of the given string.

The **last** [redacted] ways a `Null` (`'\0'`) character [redacted] till it encounters a **Null** (`'\0'`) [redacted]

In the loop, the value of `i` is incremented in every iteration when a character is counted. Finally, we print the value of `i` as the `length` of the string.

```c
#include<stdio.h>
void main() {
	char ch[20];
	int i;
	printf("Enter a string : ");
	scanf("%s",ch); //Complete the statement
	for (i = 0; ch[i] != '\0'; i++) { //Write the initializtion, condition and increment part in for loop
		
	}
	printf("The length of the string %s is %d\n", ch,i); //Complete the statement
}
```



CODETANTRA
Home
lavkush.kumar.ec2025@mitmeerut.ac.in
Support
Logout
Problem Solving Lab - MPP151
Search course
ctrl + k
30.2.2. Individual Characters of a String in
30.2.3. Count Total Number of Words in a
30.2.4. Replace Spaces in a String with a ...
30.2.5. Equal Or Not
31. Lecture 31
31.1. Fundamentals of Strings
31.1.1. Reading and displaying a string usi...
31.1.2. Reading and displaying a string usi
31.1.3. Reading a string using getchar()
31.1.4. Program to find length of a string
31.1.5. Fill in the missing code
31.1.6. Problem Solving
31.1.7. Problem Solving
31.2. Practice and Scenario Based Programs
32. Lecture 32
33. Lecture 33
34. Lecture 34
35. Lecture 35
36. Lecture 36
37. Lecture 37
38. Lecture 38
39. Lecture 39
40. Lecture 40
Explorer
C StringLo...
Submit
Debugger
Sample Test Cases
< Prev
Reset
Submit
Next >

31.1.6. Problem Solving

Fill in the missing code in the below sample code to convert all `uppercase` characters of a given string into `lowercase` characters

The **ASCII** value of `'A'` is `65` while `'a'` is `97`, the difference between them is `32`.

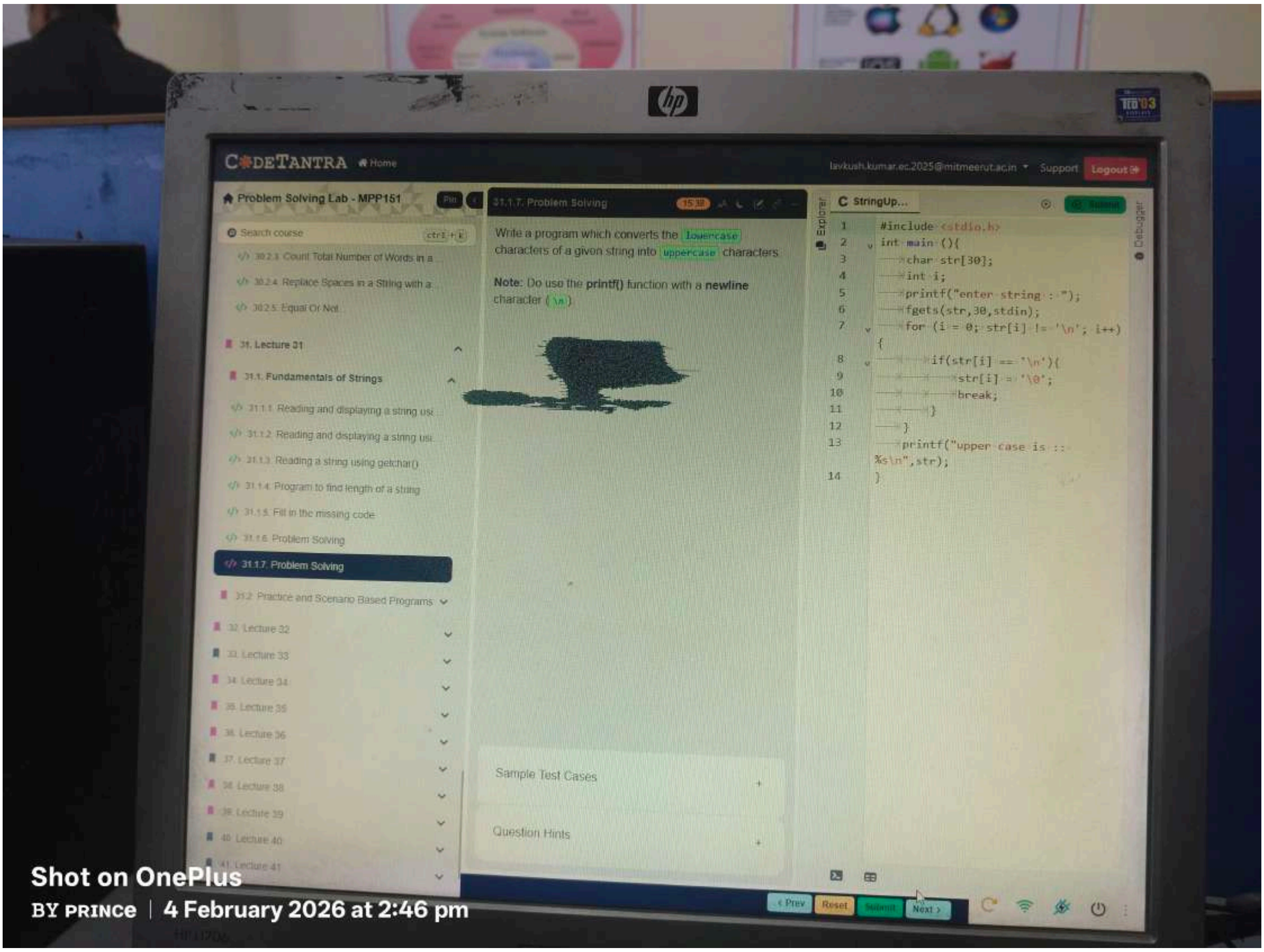
So, addin[redacted] of `'A'` becomes `'a'`.

[redacted] variable `i` to `0`, use a **loop** to iterate from **first** character to the **last** character of a string

As the **last** character in a string is always a `null` (`'\0'`) character, the condition check is on the **Null** (`'\0'`) character

Inside the loop, write the selective statement `if` to check if the character is **uppercase** or not, if it is an uppercase character then convert it to lowercase else keep the character as it is.

```c
#include<stdio.h>
void main() {
	char str[30];
	int i;
	printf("Enter any string : ");
	scanf("%s", str);
	printf("The given string is : %s\n",str); //Complete the statement
	for (i = 0; str[i] != '\0'; i++) { //Complete the code in for
			if(str[i] >= 'A' && str[i] <= 'Z') { //Write the condition to check the character is upper or not
				str[i] = str[i] + 32; //Complete the statement
			}
	}
	printf("The string in lower case is : %s\n",str); //Complete the statement
}
```

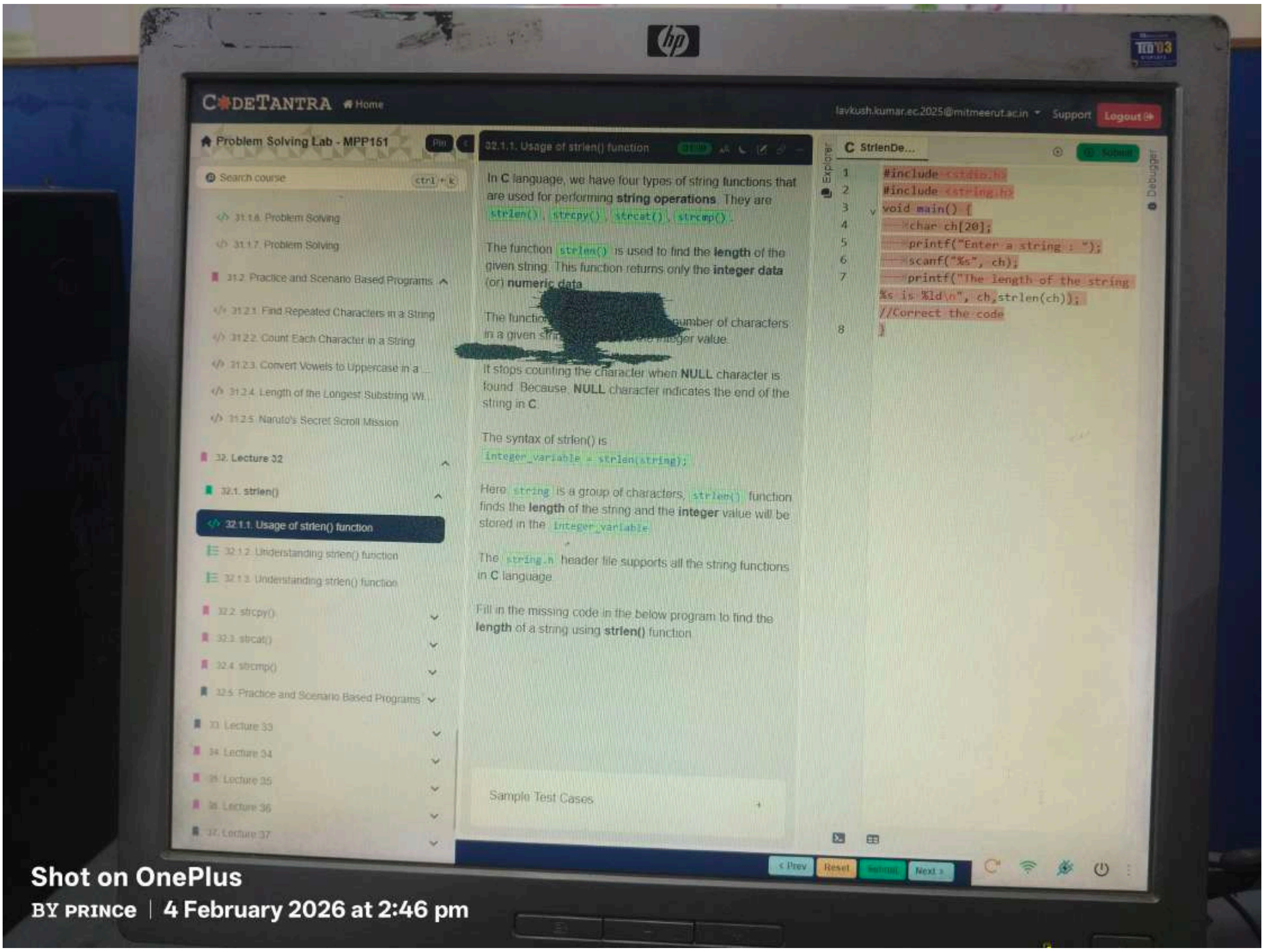

CODETANTRA
Home
lavkush.kumar.ec.2025@mitmeerut.ac.in
Support
Logout
Problem Solving Lab - MPP151
Search course
ctrl + k
30.2.3. Count Total Number of Words in a ...
30.2.4. Replace Spaces in a String with a ...
30.2.5. Equal Or Not
31. Lecture 31
31.1. Fundamentals of Strings
31.1.1. Reading and displaying a string usi...
31.1.2. Reading and displaying a string usi...
31.1.3. Reading a string using getchar()
31.1.4. Program to find length of a string
31.1.5. Fill in the missing code
31.1.6. Problem Solving
31.1.7. Problem Solving
31.2. Practice and Scenario Based Programs
32. Lecture 32
33. Lecture 33
34. Lecture 34
35. Lecture 35
36. Lecture 36
37. Lecture 37
38. Lecture 38
39. Lecture 39
40. Lecture 40
41. Lecture 41
31.1.7. Problem Solving
15:38
Sample Test Cases
Question Hints
Explorer
StringUp...
Submit
Debugger
< Prev
Reset
Submit
Next >

Write a program which converts the `lowercase` characters of a givon string into `uppercase` characters.

Note: Do use the **printf()** function with a **newline** character (`\n`).

```c
#include <stdio.h>
int main (){
    char str[30];
    int i;
    printf("enter string : ");
    fgets(str,30,stdin);
    for (i = 0; str[i] != '\n'; i++) {
            if(str[i] == '\n'){
                str[i] = '\0';
                break;
            }
    }
    printf("upper case is :: %s\n",str);
}
```

Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:46 pm



CODETANTRA
Home
lavkush.kumar.ec.2025@mitmeerut.ac.in
Support
Logout
Problem Solving Lab - MPP151
Pin
Search course
ctrl + k
31.1.6. Problem Solving
31.1.7. Problem Solving
31.2. Practice and Scenario Based Programs
31.2.1. Find Repeated Characters in a String
31.2.2. Count Each Character in a String
31.2.3. Convert Vowels to Uppercase in a ...
31.2.4. Length of the Longest Substring Wi...
31.2.5. Naruto's Secret Scroll Mission
32. Lecture 32
32.1. strlen()
32.1.1. Usage of strlen() function
32.1.2. Understanding strlen() function
32.1.3. Understanding strlen() function
32.2. strcpy()
32.3. strcat()
32.4. strcmp()
32.5. Practice and Scenario Based Programs
33. Lecture 33
34. Lecture 34
35. Lecture 35
36. Lecture 36
37. Lecture 37
32.1.1. Usage of strlen() function
Explorer
C StrlenDe...
Submit
Debugger
Sample Test Cases
< Prev
Reset
Submit
Next >

In **C** language, we have four types of string functions that are used for performing **string operations**. They are `strlen()`, `strcpy()`, `strcat()`, `strcmp()`.

The function `strlen()` is used to find the **length** of the given string. This function returns only the **integer data** (or) **numeric data**.

The functio[redacted]number of characters in a given stri[redacted]ger value.

It stops counting the character when **NULL** character is found. Because, **NULL** character indicates the end of the string in **C**.

The syntax of strlen() is

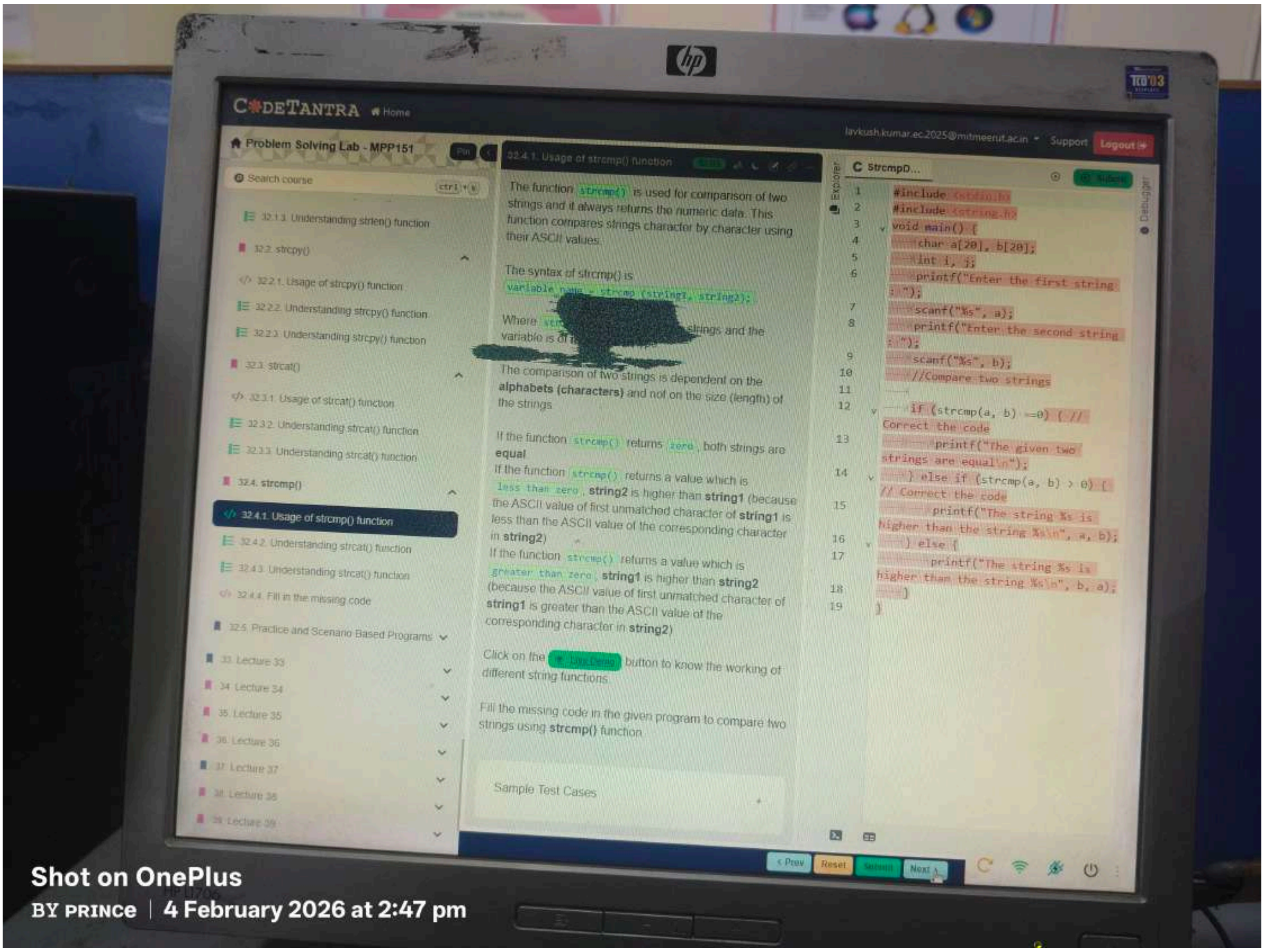
`integer_variable = strlen(string);`

Here `string` is a group of characters, `strlen()` function finds the **length** of the string and the **integer** value will be stored in the `integer_variable`.

The `string.h` header file supports all the string functions in **C** language.

Fill in the missing code in the below program to find the **length** of a string using **strlen()** function.

```c
#include <stdio.h>
#include <string.h>
void main() {
	char ch[20];
	printf("Enter a string : ");
	scanf("%s", ch);
	printf("The length of the string %s is %ld\n", ch,strlen(ch)); //Correct the code
}
```



CODETANTRA Home

Problem Solving Lab - MPP151

Search course

- 32.1.3. Understanding strlen() function
- 32.2. strcpy()
- 32.2.1. Usage of strcpy() function
- 32.2.2. Understanding strcpy() function
- 32.2.3. Understanding strcpy() function
- 32.3. strcat()
- 32.3.1. Usage of strcat() function
- 32.3.2. Understanding strcat() function
- 32.3.3. Understanding strcat() function
- **32.4. strcmp()**
- **32.4.1. Usage of strcmp() function**
- 32.4.2. Understanding strcat() function
- 32.4.3. Understanding strcat() function
- 32.4.4. Fill in the missing code
- 32.5. Practice and Scenario Based Programs
- 33. Lecture 33
- 34. Lecture 34
- 35. Lecture 35
- 36. Lecture 36
- 37. Lecture 37
- 38. Lecture 38
- 39. Lecture 39

32.4.1. Usage of strcmp() function

The function `strcmp()` is used for comparison of two strings and it always returns the numeric data. This function compares strings character by character using their ASCII values.

The syntax of strcmp() is

`variable_name = strcmp (string1, string2);`

Where [redacted] strings and the variable is of

The comparison of two strings is dependent on the **alphabets (characters)** and not on the size (length) of the strings

If the function `strcmp()` returns `zero`, both strings are **equal**

If the function `strcmp()` returns a value which is `less than zero`, **string2** is higher than **string1** (because the ASCII value of first unmatched character of **string1** is less than the ASCII value of the corresponding character in **string2**)

If the function `strcmp()` returns a value which is `greater than zero`, **string1** is higher than **string2** (because the ASCII value of first unmatched character of **string1** is greater than the ASCII value of the corresponding character in **string2**)

Click on the Live Demo button to know the working of different string functions

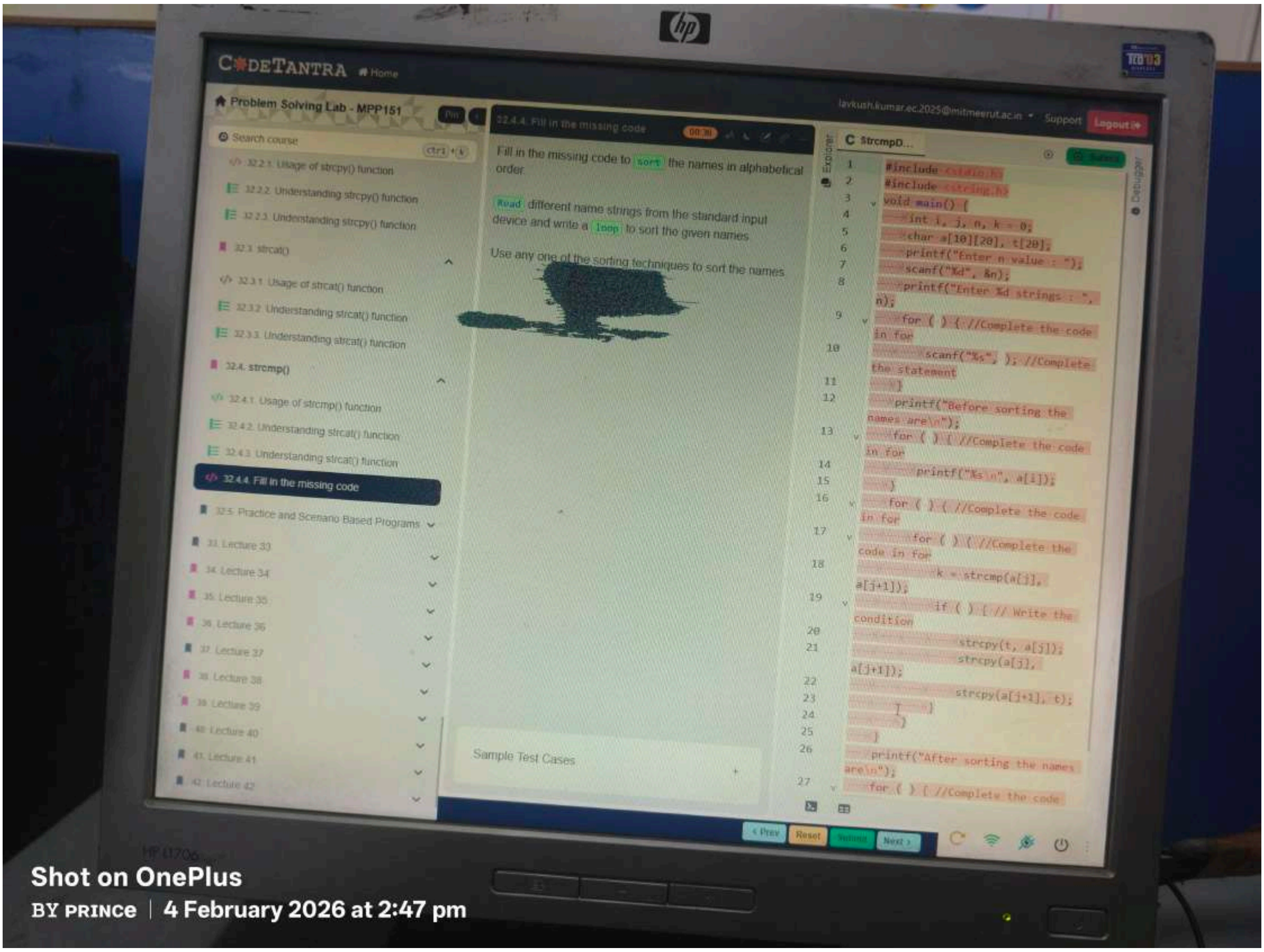
Fill the missing code in the given program to compare two strings using **strcmp()** function

Sample Test Cases

laykush.kumar.ec.2025@mitmeerut.ac.in Support Logout

C StrcmpD...

```
#include <stdio.h>
#include <string.h>
void main() {
	char a[20], b[20];
	int i, j;
	printf("Enter the first string : ");
	scanf("%s", a);
	printf("Enter the second string : ");
	scanf("%s", b);
	//Compare two strings
	
	if (strcmp(a, b) ==0) { // Correct the code
		printf("The given two strings are equal\n");
	} else if (strcmp(a, b) > 0) { // Correct the code
		printf("The string %s is higher than the string %s\n", a, b);
	} else {
		printf("The string %s is higher than the string %s\n", b, a);
	}
}
```


CODETANTRA
Home
Problem Solving Lab - MPP151
Pin
Search course
ctrl + k
32.2.1. Usage of strcpy() function
32.2.2. Understanding strcpy() function
32.2.3. Understanding strcpy() function
32.3. strcat()
32.3.1. Usage of strcat() function
32.3.2. Understanding strcat() function
32.3.3. Understanding strcat() function
32.4. strcmp()
32.4.1. Usage of strcmp() function
32.4.2. Understanding strcat() function
32.4.3. Understanding strcat() function
32.4.4. Fill in the missing code
32.5. Practice and Scenario Based Programs
33. Lecture 33
34. Lecture 34
35. Lecture 35
36. Lecture 36
37. Lecture 37
38. Lecture 38
39. Lecture 39
40. Lecture 40
41. Lecture 41
42. Lecture 42
laykush.kumar.ec2025@mitmeerut.ac.in
Support
Logout
Explorer
StrcmpD...
Submit
Debugger
Sample Test Cases
< Prev
Reset
Submit
Next >

32.4.4. Fill in the missing code

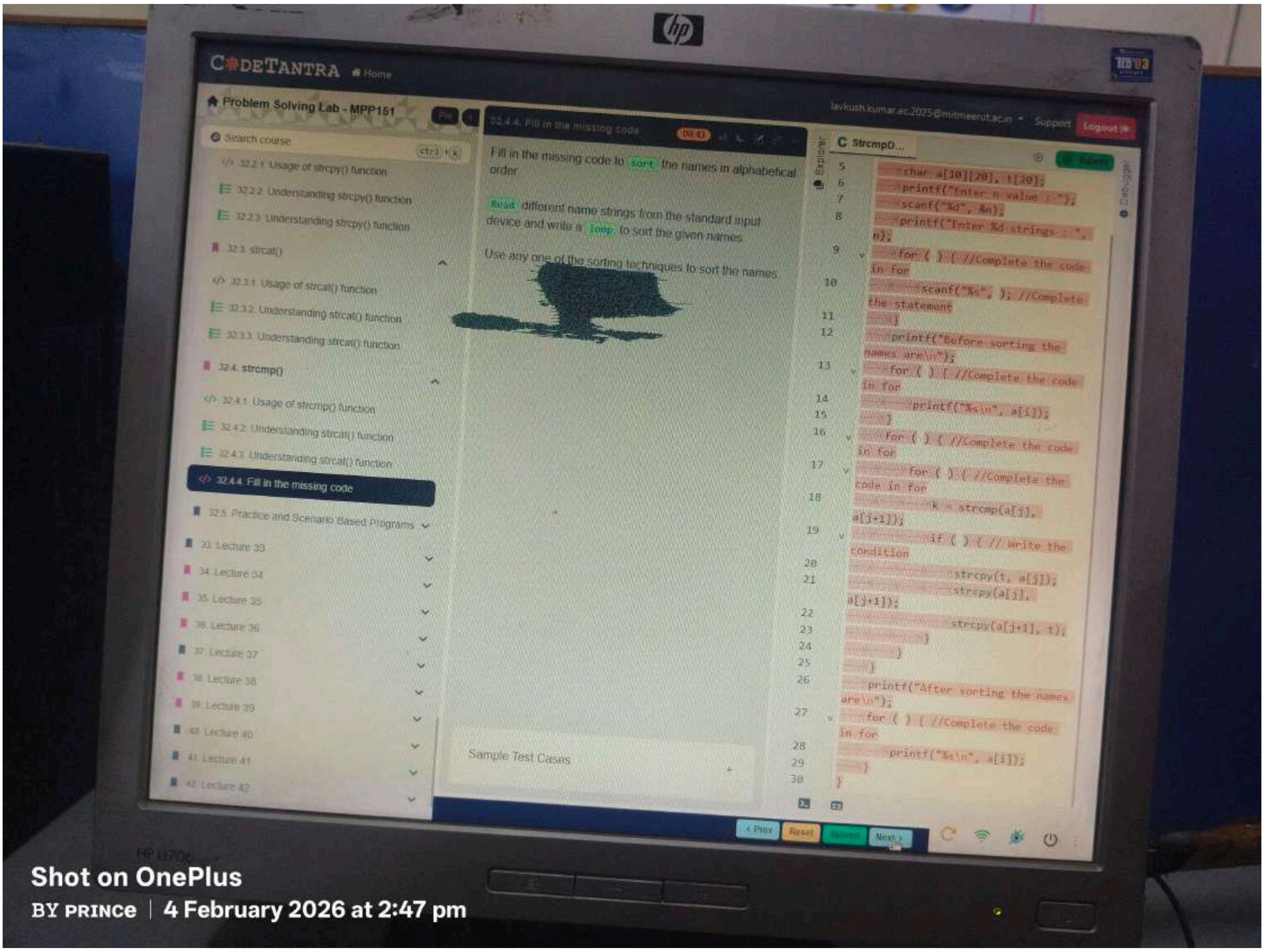
Fill in the missing code to `sort` the names in alphabetical order.

`Read` different name strings from the standard input device and write a `loop` to sort the given names.

Use any one of the sorting techniques to sort the names

[redacted]

```c
#include <stdio.h>
#include <string.h>
void main() {
	int i, j, n, k = 0;
	char a[10][20], t[20];
	printf("Enter n value : ");
	scanf("%d", &n);
	printf("Enter %d strings : ", n);
	for ( ) { //Complete the code in for
		scanf("%s", ); //Complete the statement
	}
	printf("Before sorting the names are\n");
	for ( ) { //Complete the code in for
		printf("%s\n", a[i]);
	}
	for ( ) { //Complete the code in for
		for ( ) { //Complete the code in for
			k = strcmp(a[j], a[j+1]);
			if ( ) { // Write the condition
				strcpy(t, a[j]);
				strcpy(a[j], a[j+1]);
				strcpy(a[j+1], t);
			}
		}
	}
	printf("After sorting the names are\n");
	for ( ) { //Complete the code
```

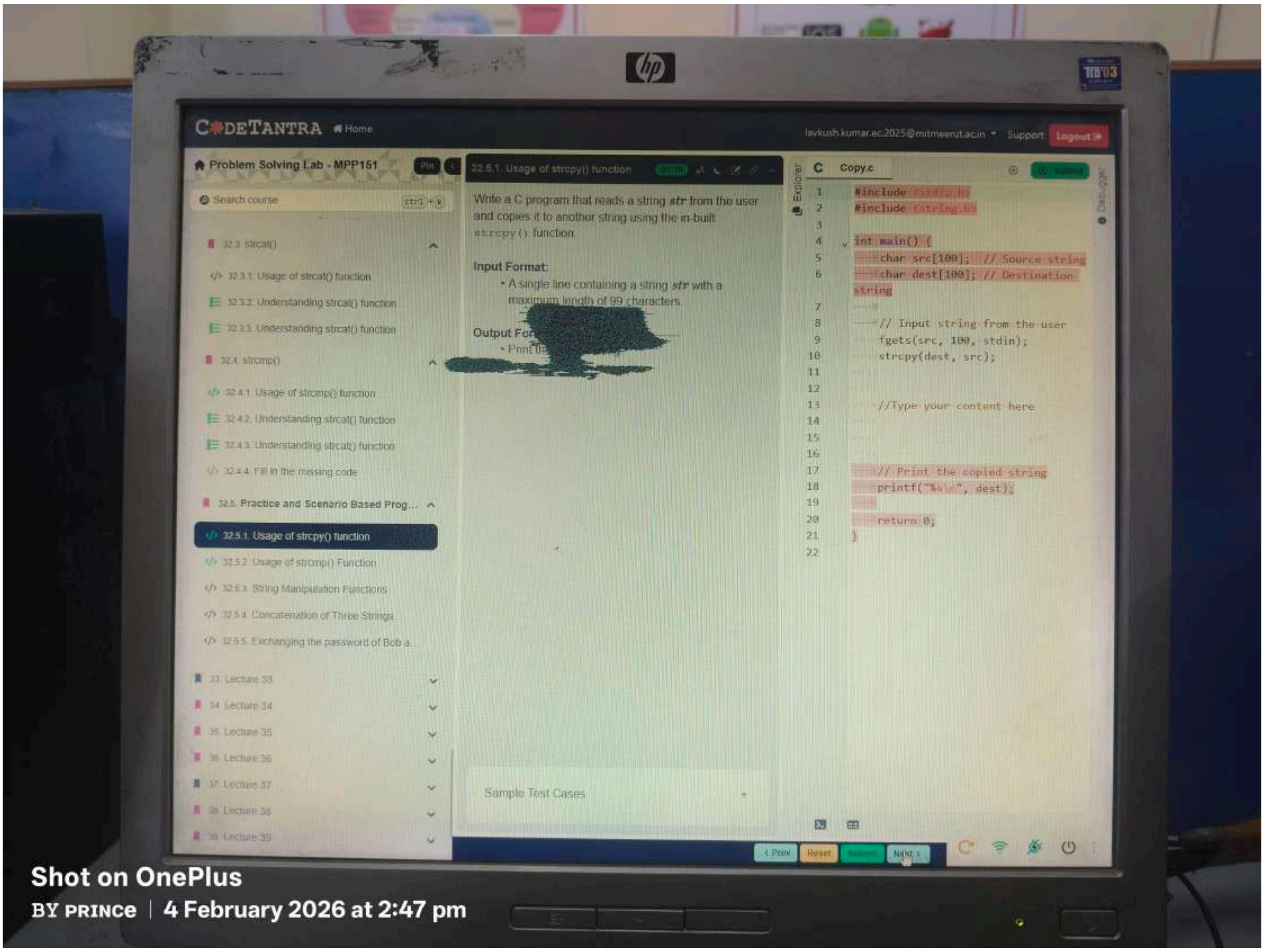

CODETANTRA
Home
Problem Solving Lab - MPP151
Search course
ctrl + k
32.2.1. Usage of strcpy() function
32.2.2. Understanding strcpy() function
32.2.3. Understanding strcpy() function
32.3. strcat()
32.3.1. Usage of strcat() function
32.3.2. Understanding strcat() function
32.3.3. Understanding strcat() function
32.4. strcmp()
32.4.1. Usage of strcmp() function
32.4.2. Understanding strcat() function
32.4.3. Understanding strcat() function
32.4.4. Fill in the missing code
32.5. Practice and Scenario Based Programs
33. Lecture 33
34. Lecture 34
35. Lecture 35
36. Lecture 36
37. Lecture 37
38. Lecture 38
39. Lecture 39
40. Lecture 40
41. Lecture 41
42. Lecture 42
32.4.4. Fill in the missing code
lavkush.kumar.ec.2025@mitmeerut.ac.in
Support
Logout
Explorer
C StrcmpD...
Submit
Debugger
Sample Test Cases
+
< Prev
Reset
Submit
Next >
Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:47 pm

Fill in the missing code to sort the names in alphabetical order

Read different name strings from the standard input device and write a loop to sort the given names

Use any one of the sorting techniques to sort the names

```
	char a[10][20], t[20];
	printf("Enter n value : ");
	scanf("%d", &n);
	printf("Enter %d strings : ", n);
	for ( ) { //Complete the code in for
		scanf("%s", ); //Complete the statement
	}
	printf("Before sorting the names are\n");
	for ( ) { //Complete the code in for
		printf("%s\n", a[i]);
	}
	for ( ) { //Complete the code in for
		for ( ) { //Complete the code in for
			k = strcmp(a[j], a[j+1]);
			if ( ) { // write the condition
				strcpy(t, a[j]);
				strcpy(a[j], a[j+1]);
				strcpy(a[j+1], t);
			}
		}
	}
	printf("After sorting the names are\n");
	for ( ) { //Complete the code in for
		printf("%s\n", a[i]);
	}
}
```


CODETANTRA
Home
lavkush.kumar.ec.2025@mitmeerut.ac.in
Support
Logout
Problem Solving Lab - MPP151
Pin
Search course
ctrl + k
32.3. strcat()
32.3.1. Usage of strcat() function
32.3.2. Understanding strcat() function
32.3.3. Understanding strcat() function
32.4. strcmp()
32.4.1. Usage of strcmp() function
32.4.2. Understanding strcat() function
32.4.3. Understanding strcat() function
32.4.4. Fill in the missing code
32.5. Practice and Scenario Based Prog...
32.5.1. Usage of strcpy() function
32.5.2. Usage of strcmp() Function
32.5.3. String Manipulation Functions
32.5.4. Concatenation of Three Strings
32.5.5. Exchanging the password of Bob a...
33. Lecture 33
34. Lecture 34
35. Lecture 35
36. Lecture 36
37. Lecture 37
38. Lecture 38
39. Lecture 39
Explorer
Debugger
Copy.c
Submit
Sample Test Cases
‹ Prev
Reset
Submit
Next ›

32.5.1. Usage of strcpy() function

Write a C program that reads a string *str* from the user and copies it to another string using the in-built `strcpy()` function.

Input Format:

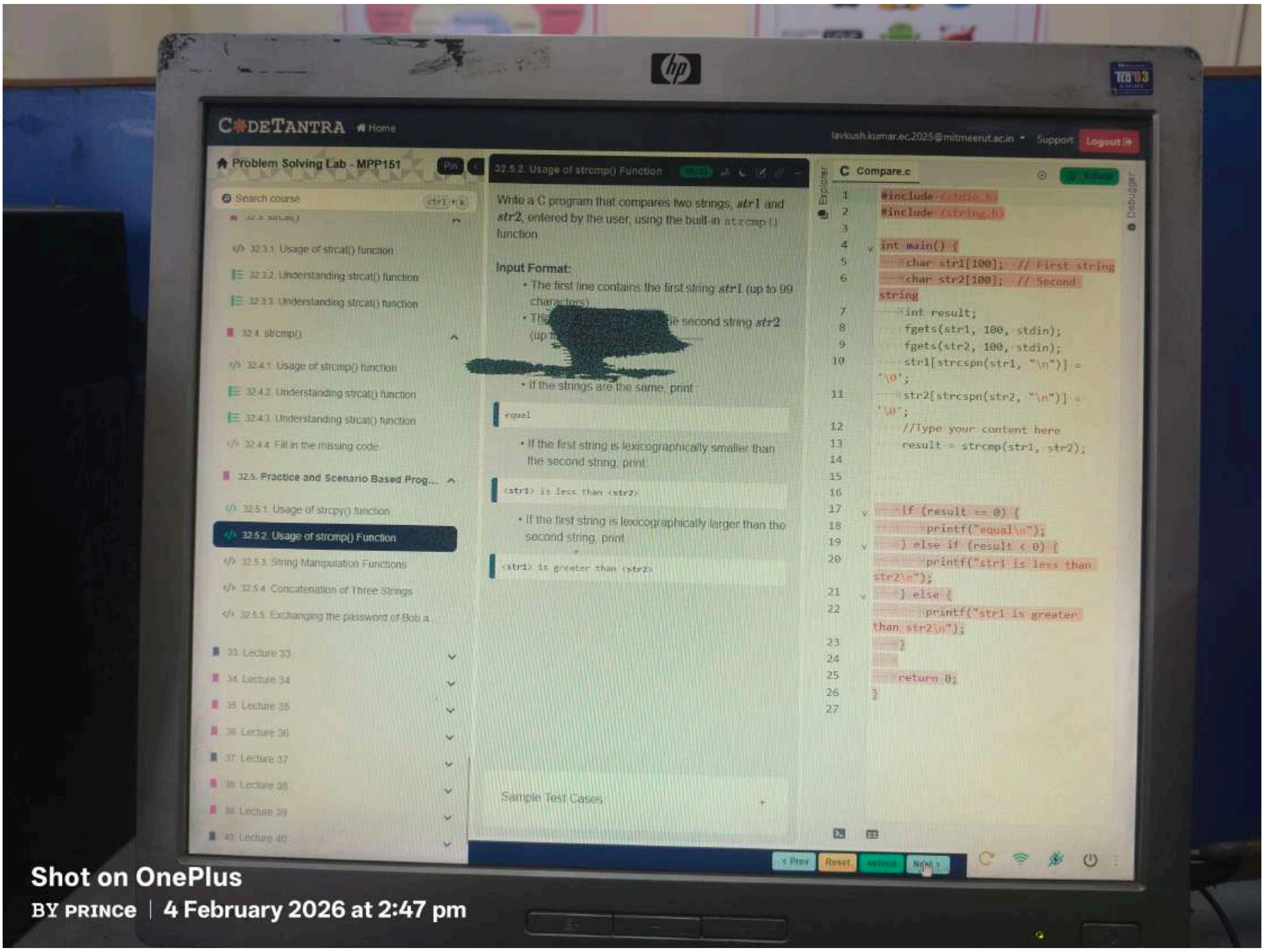
- A single line containing a string *str* with a maximum length of 99 characters.

Output For[redacted]

- Print th

```c
#include <stdio.h>
#include <string.h>

int main() {
    char src[100];  // Source string
    char dest[100]; // Destination string
    
    // Input string from the user
    fgets(src, 100, stdin);
    strcpy(dest, src);
    
    
    //Type your content here
    
    
    
    // Print the copied string
    printf("%s\n", dest);
    
    return 0;
}
```

Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:47 pm


CODETANTRA
Home
lavkush.kumar.ec.2025@mitmeerut.ac.in
Support
Logout
Problem Solving Lab - MPP151
Pin
Search course
ctrl + k
32.3.1. Usage of strcat() function
32.3.2. Understanding strcat() function
32.3.3. Understanding strcat() function
32.4. strcmp()
32.4.1. Usage of strcmp() function
32.4.2. Understanding strcat() function
32.4.3. Understanding strcat() function
32.4.4. Fill in the missing code
32.5. Practice and Scenario Based Prog...
32.5.1. Usage of strcpy() function
32.5.2. Usage of strcmp() Function
32.5.3. String Manipulation Functions
32.5.4. Concatenation of Three Strings
32.5.5. Exchanging the password of Bob a...
33. Lecture 33
34. Lecture 34
35. Lecture 35
36. Lecture 36
37. Lecture 37
38. Lecture 38
39. Lecture 39
40. Lecture 40
Explorer
Compare.c
Debugger
Sample Test Cases
< Prev
Reset
Submit
Next >
Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:47 pm

32.5.2. Usage of strcmp() Function

Write a C program that compares two strings, *str1* and *str2*, entered by the user, using the built-in `strcmp()` function

Input Format:

- The first line contains the first string *str1* (up to 99 characters)
- Th[redacted]e second string *str2* (up t

- If the strings are the same, print

```
equal
```

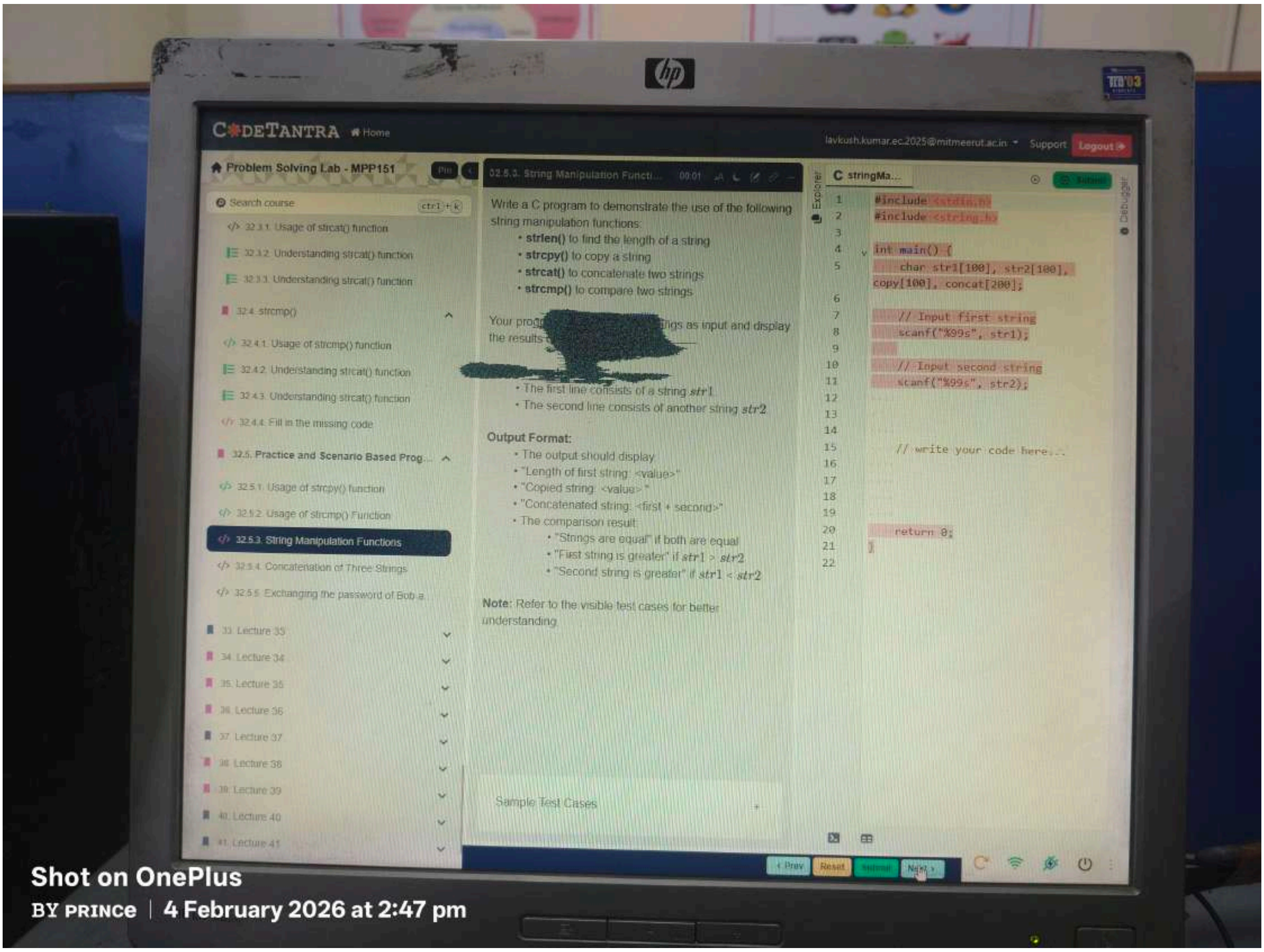
- If the first string is lexicographically smaller than the second string, print

```
<str1> is less than <str2>
```

- If the first string is lexicographically larger than the second string, print

```
<str1> is greater than <str2>
```

```c
#include <stdio.h>
#include <string.h>

int main() {
	char str1[100];  // First string
	char str2[100];  // Second string
	int result;
	fgets(str1, 100, stdin);
	fgets(str2, 100, stdin);
	str1[strcspn(str1, "\n")] = '\0';
	str2[strcspn(str2, "\n")] = '\0';
	//Type your content here
	result = strcmp(str1, str2);


	if (result == 0) {
		printf("equal\n");
	} else if (result < 0) {
		printf("str1 is less than str2\n");
	} else {
		printf("str1 is greater than str2\n");
	}

	return 0;
}
```



CODETANTRA
Home
lavkush.kumar.ec.2025@mitmeerut.ac.in
Support
Logout
Problem Solving Lab - MPP151
Pin
Search course
ctrl + k
32.3.1. Usage of strcat() function
32.3.2. Understanding strcat() function
32.3.3. Understanding strcat() function
32.4. strcmp()
32.4.1. Usage of strcmp() function
32.4.2. Understanding strcat() function
32.4.3. Understanding strcat() function
32.4.4. Fill in the missing code
32.5. Practice and Scenario Based Prog...
32.5.1. Usage of strcpy() function
32.5.2. Usage of strcmp() Function
32.5.3. String Manipulation Functions
32.5.4. Concatenation of Three Strings
32.5.5. Exchanging the password of Bob a...
33. Lecture 33
34. Lecture 34
35. Lecture 35
36. Lecture 36
37. Lecture 37
38. Lecture 38
39. Lecture 39
40. Lecture 40
41. Lecture 41
32.5.3. String Manipulation Functi...
00:01
Explorer
C stringMa...
Submit
Debugger
Sample Test Cases
‹ Prev
Reset
Submit
Next ›
Shot on OnePlus
BY PRINCE | 4 February 2026 at 2:47 pm

Write a C program to demonstrate the use of the following string manipulation functions:

- **strlen()** to find the length of a string
- **strcpy()** to copy a string
- **strcat()** to concatenate two strings
- **strcmp()** to compare two strings

Your prog[redacted]ngs as input and display the results

- The first line consists of a string $str1$
- The second line consists of another string $str2$

Output Format:

- The output should display
- "Length of first string: <value>"
- "Copied string: <value> "
- "Concatenated string: <first + second>"
- The comparison result:
  - "Strings are equal" if both are equal
  - "First string is greater" if $str1 > str2$
  - "Second string is greater" if $str1 < str2$

Note: Refer to the visible test cases for better understanding.

```c
#include <stdio.h>
#include <string.h>

int main() {
    char str1[100], str2[100], copy[100], concat[200];

    // Input first string
    scanf("%99s", str1);

    // Input second string
    scanf("%99s", str2);



    // write your code here...




    return 0;
}
```