Week 03

**Question 1:**

**Write a program to read two integer values and print true if both the numbers end with the same digit, otherwise print false. Example: If 698 and 768 are given, program should print true as they both end with 8.**

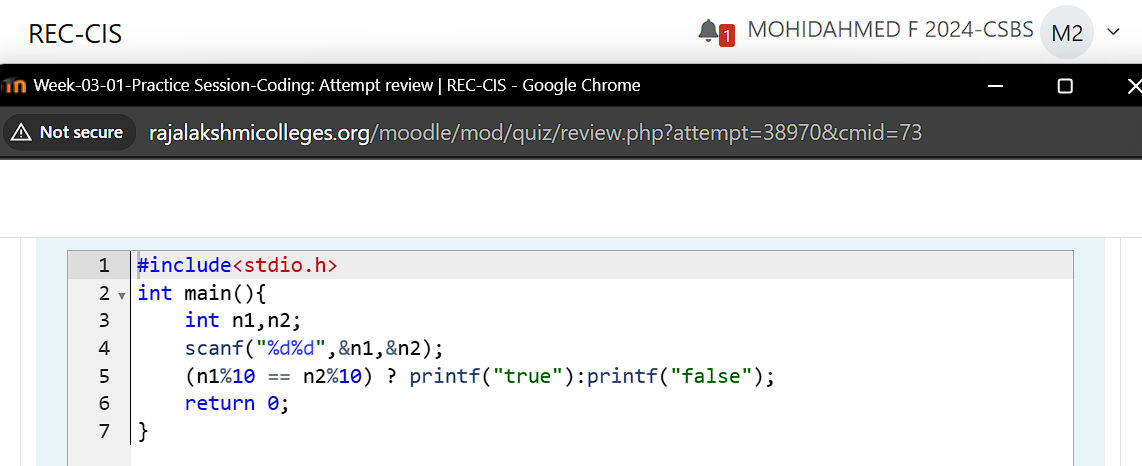
**Sample Input 1:**

**25 53**

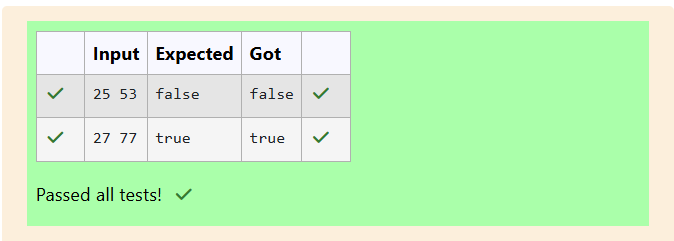
**Sample Output 1:**

**false**

**Program:**

****

**Output:**

****

**Question 2:**

**Given an integer, *n*, perform the following conditional actions:**

**·         If *n* is odd, print Weird**

**·         If *n* is even and in the inclusive range of *2* to *5*, print *Not Weird***

**·         If *n* is even and in the inclusive range of *6* to *20*, print *Weird***

**·         If *n* is even and greater than *20*, print *Not Weird***

**Input Format: A single line containing a positive integer, *n*.**

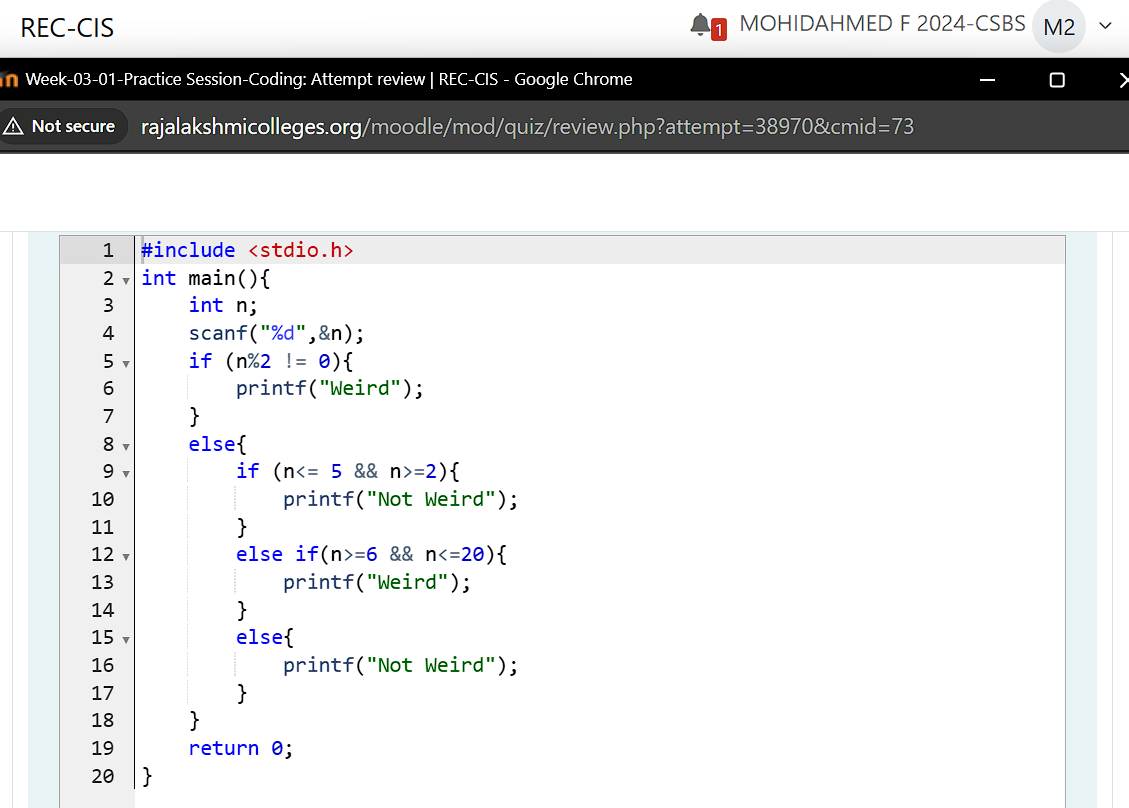
**Sample Input 1:**

**3**

**Sample Output 1:**

**Weird**

**Program:**

****

**Output:**

****

**Question 3:**

**Three numbers form a Pythagorean triple if the sum of squares of two numbers is equal to the square of the third. For example, 3, 5 and 4 form a Pythagorean triple, since 3\*3 + 4\*4 = 25 = 5\*5 You are given three integers, a, b, and c. They need not be given in increasing order. If they form a Pythagorean triple, then print "yes", otherwise, print "no". Please note that the output message is in small letters.**

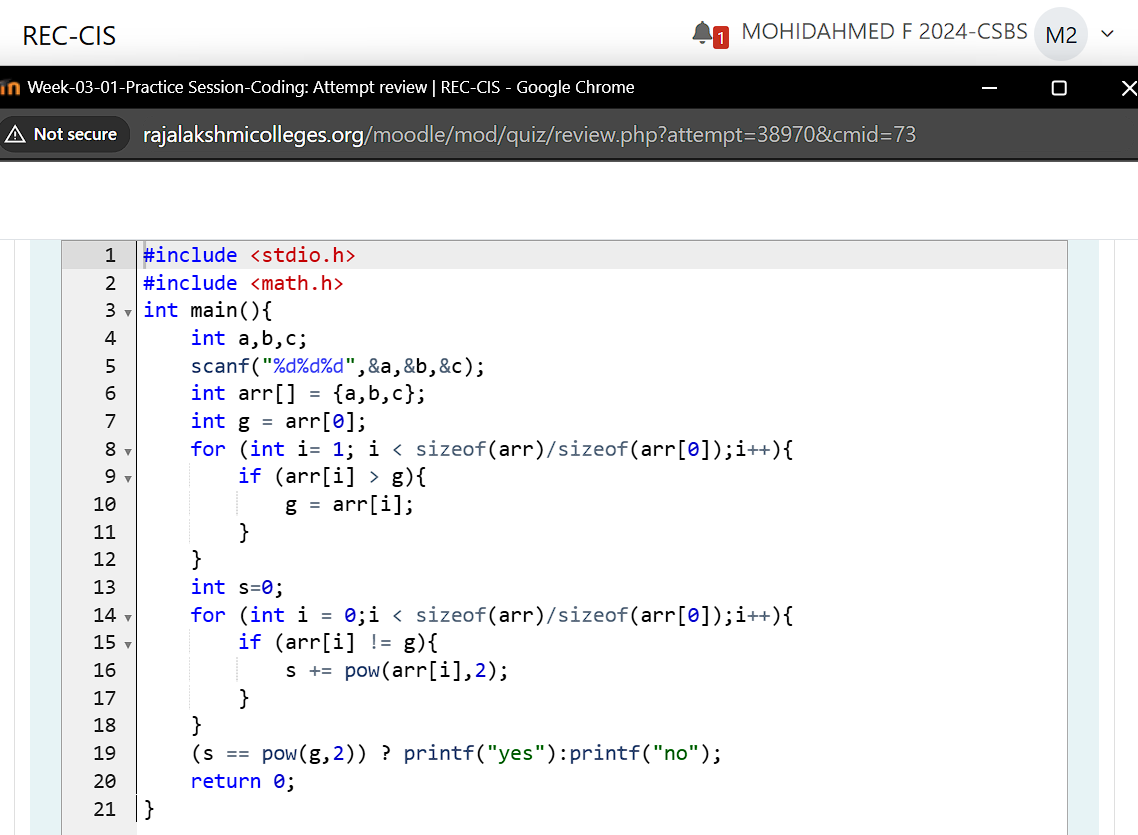
**Sample Input 1:**

**3 5 4**

**Sample Output 1:**

**yes**

**Program:**

****

**Output:**

****

**Question 4:**

**Write a program that determines the name of a shape from its number of sides. Read the number of sides from the user and then report the appropriate name as part of a meaningful message. Your program should support shapes with anywhere from 3 up to (and including) 10 sides. If a number of sides outside of this range is entered then your program should display an appropriate error message.**

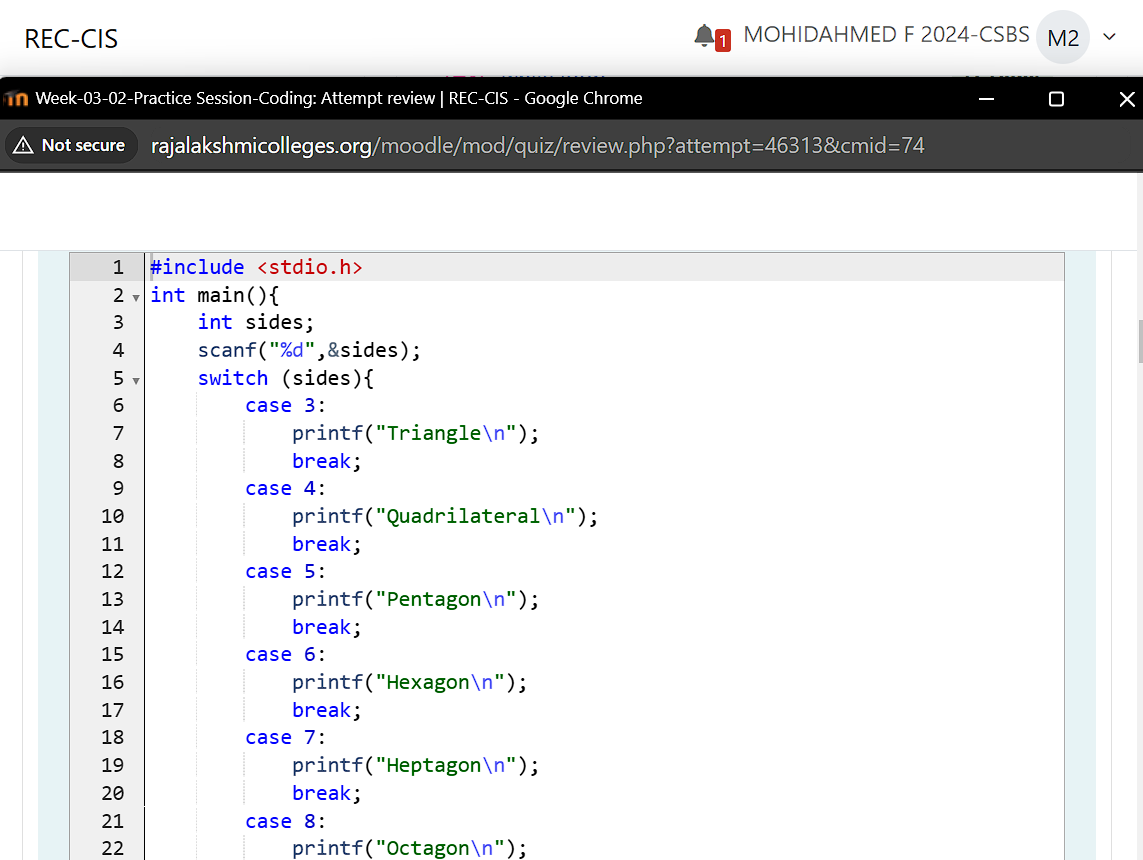
**Sample Input 1:**

**3**

**Sample Output 1:**

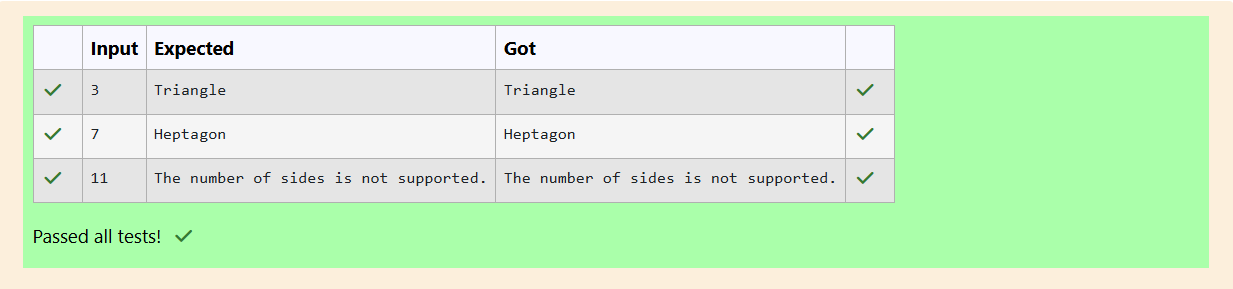
**Triangle**

**Program:**

****

****

**Output:**

****

**Question 5:**

**The Chinese zodiac assigns animals to years in a 12-year cycle. One 12-year cycle is shown in the table below. The pattern repeats from there, with 2012 being another year of the Dragon, and 1999 being another year of the Hare.**

**2000-Dragon 2001-Snake 2002-Horse 2003-Sheep 2004-Monkey 2005-Rooster 2006-Dog 2007-Pig 2008-Rat 2009-Ox 2010-Tiger 2011-Hare**

**Write a program that reads a year from the user and displays the animal associated with that year. Your program should work correctly for any year greater than or equal to zero, not just the ones listed in the table.**

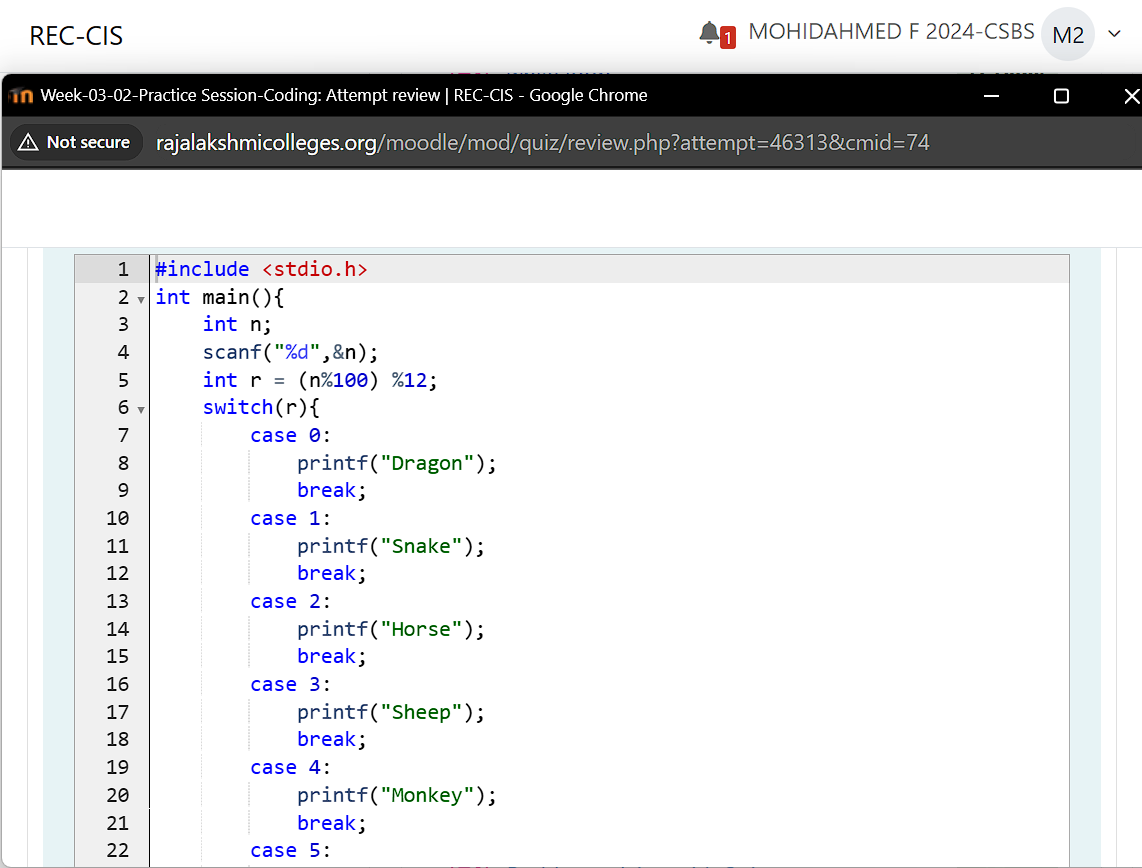
**Sample Input 1:**

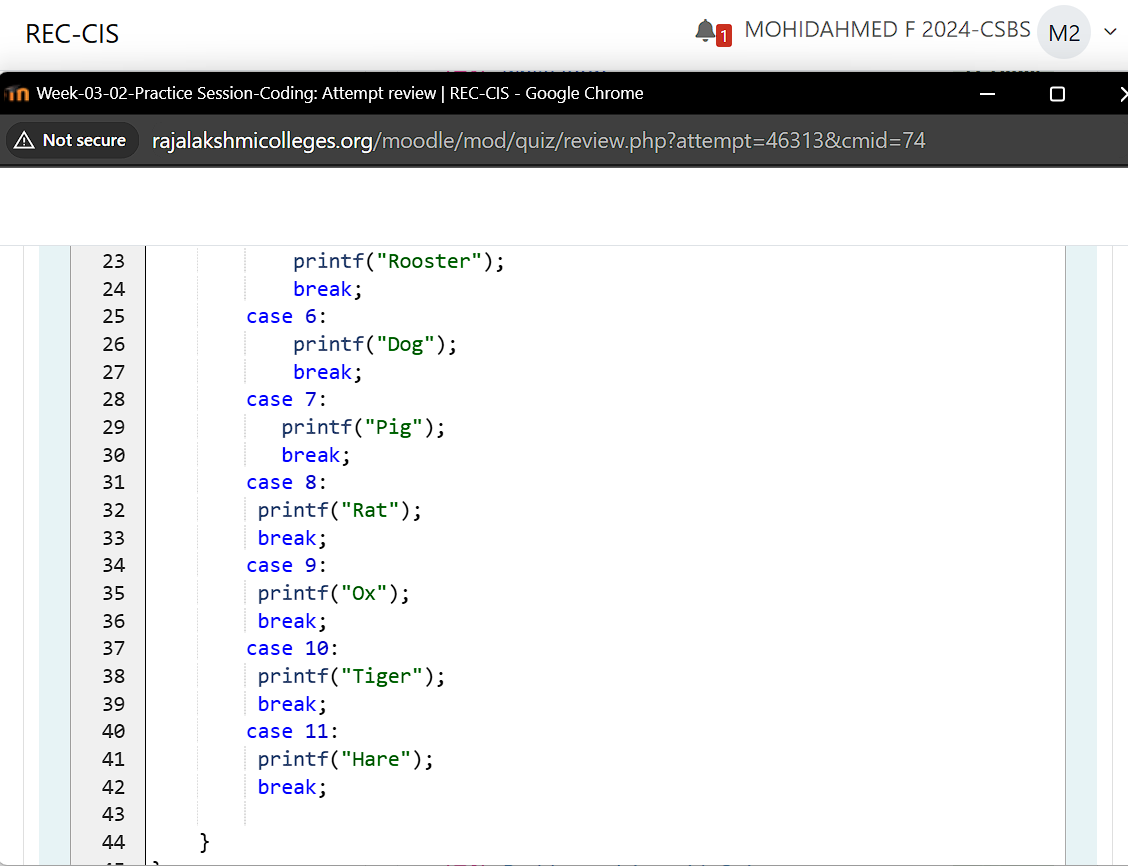
**2004**

**Sample Output 1:**

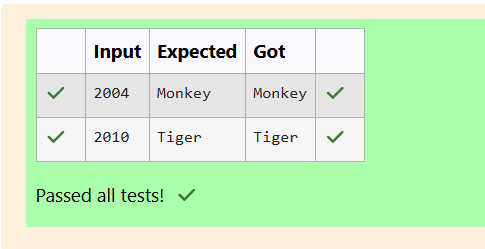
**Monkey**

**Program:**

****

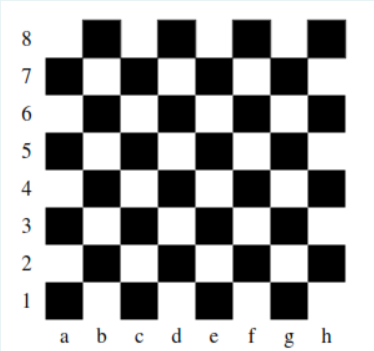
****

**Output:**

****

**Question 6:**

**Positions on a chess board are identiﬁed by a letter and a number. The letter identiﬁes the column, while the number identiﬁes the row, as shown below:**

****

**Write a program that reads a position from the user. Use an if statement to determine if the column begins with a black square or a white square. Then use modular arithmetic to report the color of the square in that row. For example, if the user enters a1 then your program should report that the square is black. If the user enters d5 then your program should report that the square is white. Your program may assume that a valid position will always be entered. It does not need to perform any error checking.**

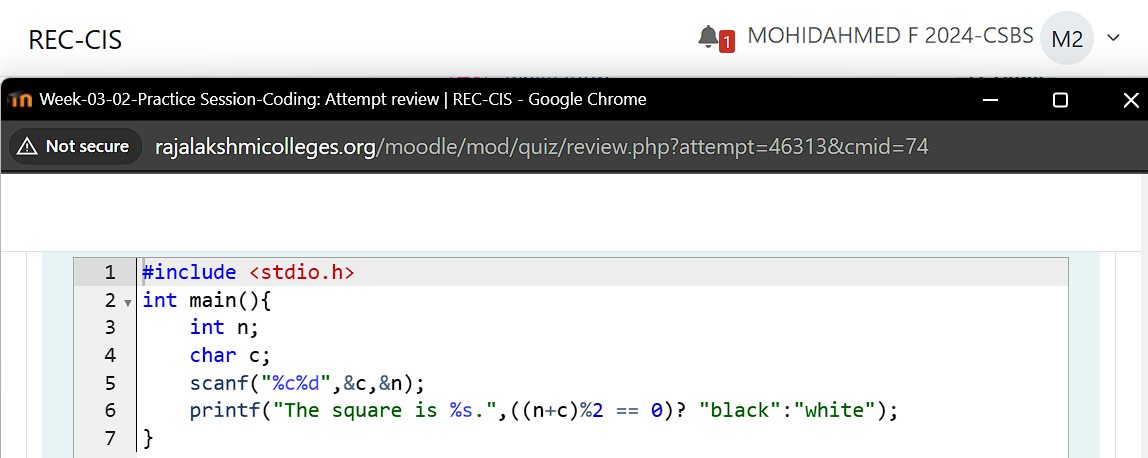
**Sample Input 1:**

**a 1**

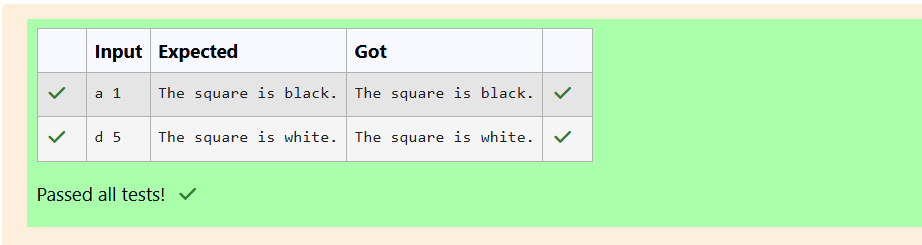
**Sample Output 1:**

**The square is black.**

**Program:**

****

**Output:**

****

**Question 7:**

**Some data sets specify dates using the year and day of year rather than the year, month, and day of month. The day of year (DOY) is the sequential day number starting with day 1 on January 1st. There are two calendars - one for normal years with 365 days, and one for leap years with 366 days. Leap years are divisible by 4. Centuries, like 1900, are not leap years unless they are divisible by 400. So, 2000 was a leap year. To find the day of year number for a standard date, scan down the Jan column to find the day of month, then scan across to the appropriate month column and read the day of year number. Reverse the process to find the standard date for a given day of year. Write a program to print the Day of Year of a given date, month and year.**

**Sample Input 1**

**18**

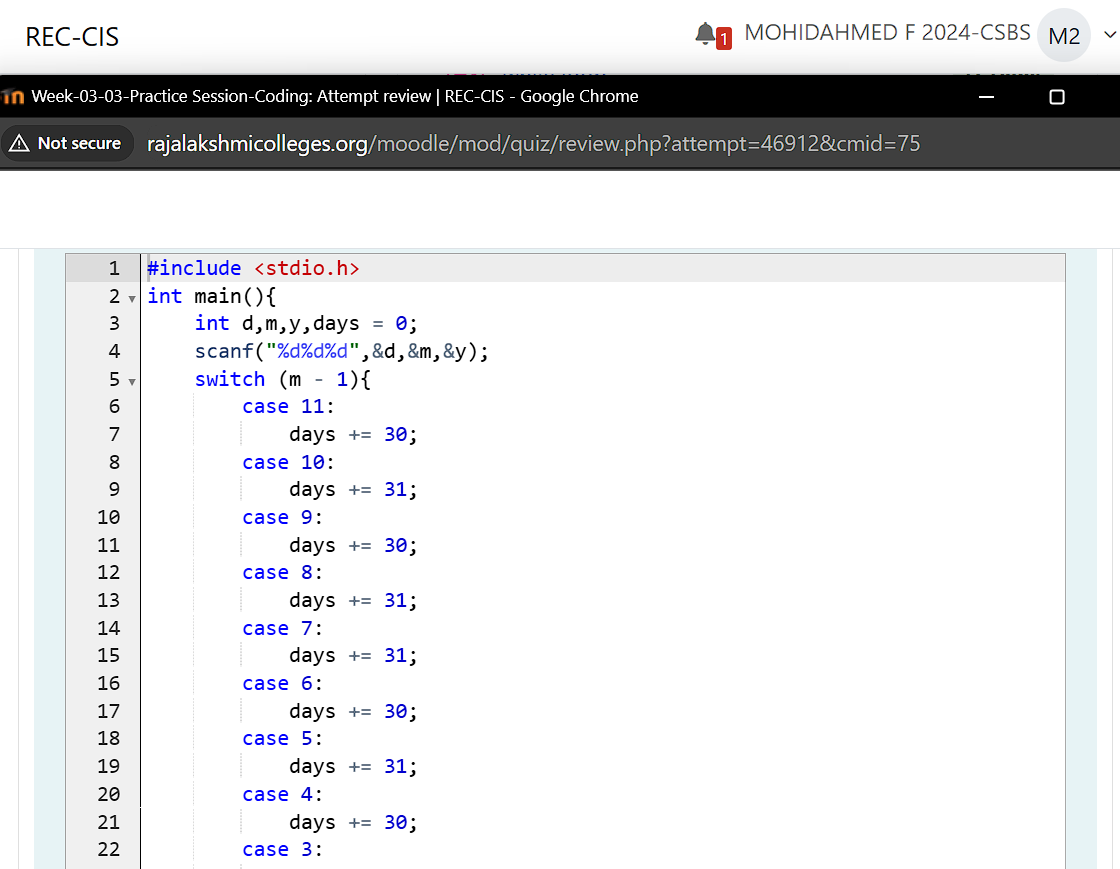
**6**

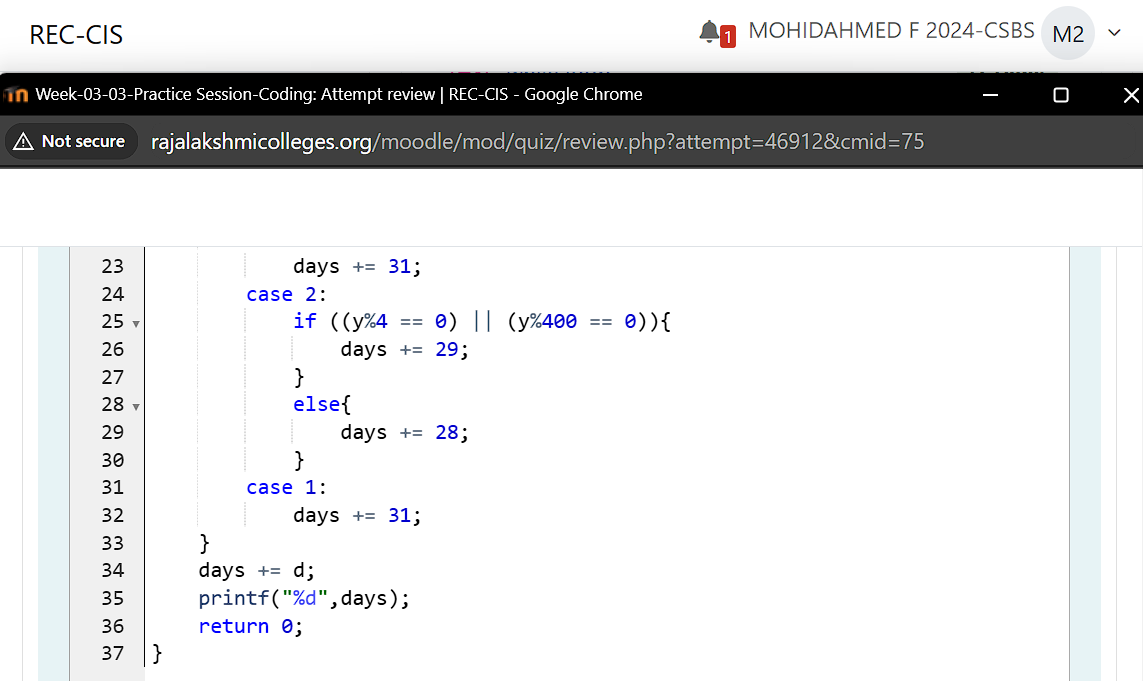
**2020**

**Sample Output 1**

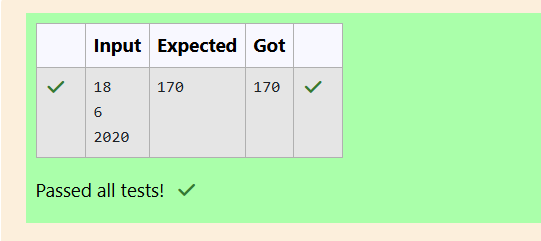
**170**

**Program:**

****

****

**Output:**



**Question 8:**

**Suppandi is trying to take part in the local village math quiz. In the first round, he is asked about shapes and areas. Suppandi, is confused, he was never any good at math. And also, he is bad at remembering the names of shapes. Instead, you will be helping him calculate the area of shapes. When he says rectangle he is actually referring to a square. When he says square, he is actually referring to a triangle. When he says triangle he is referring to a rectangle. And when he is confused, he just says something random. At this point, all you can do is say 0. Help Suppandi by printing the correct answer in an integer.**

**Input Format:**

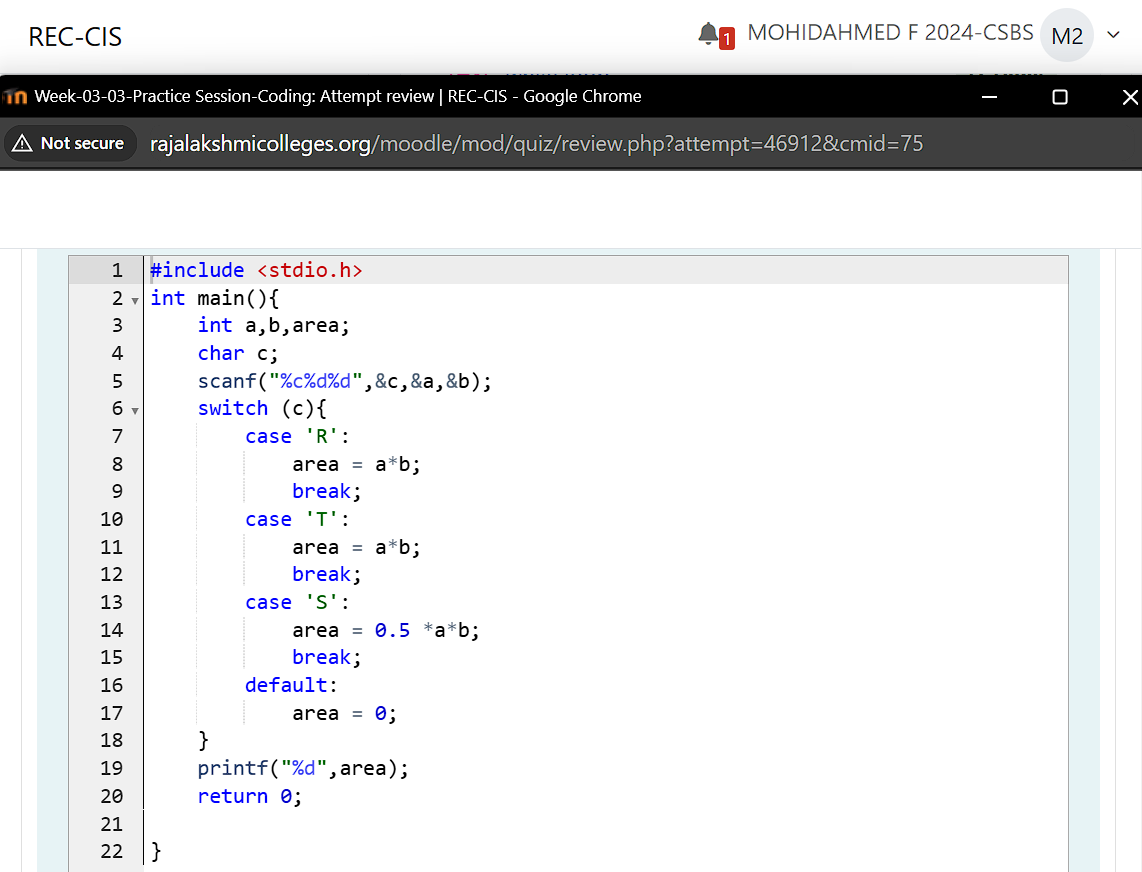
**Name of shape, Length of 1 side, Length of other side**

**Output Format: Print the area of the shape.**

**Sample Input 1: T 10 20**

**Sample Output 1: 200**

**Program:**

****

**Output:**

****

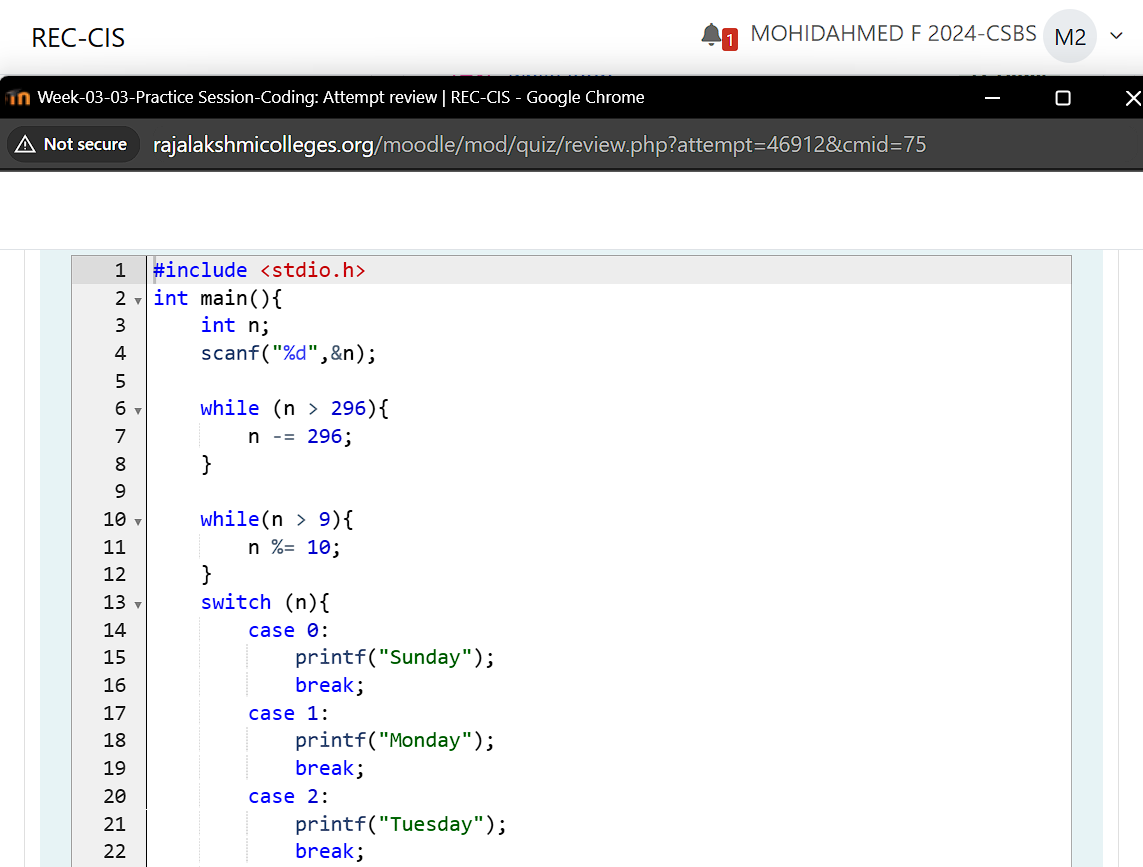
**Question 9:**

**Superman is planning a journey to his home planet. It is very important for him to know which day he arrives there. They don't follow the 7-day week like us. Instead, they follow a 10-day week with the following days: Day Number Name of Day 1 Sunday 2 Monday 3 Tuesday 4 Wednesday 5 Thursday 6 Friday 7 Saturday 8 Kryptonday 9 Coluday 10 Daxamday Here are the rules of the calendar: The calendar starts with Sunday always. It has only 296 days. After the 296th day, it goes back to Sunday. You begin your journey on a Sunday and will reach after n. You have to tell on which day you will arrive when you reach there.**

**Sample Input: 7**

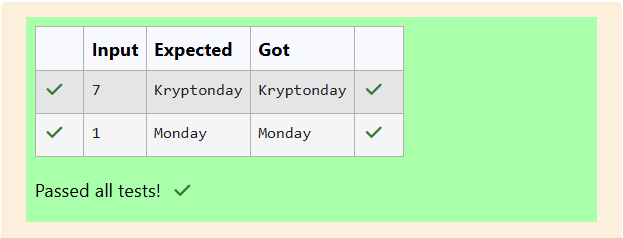
**Example Output: Kryptonday**

**Program:**

****

****

**Output:**

****

**Week 03 Assessments**

**Question 1:**

**Write a C program to find the eligibility of admission for a professional course based on the following criteria:**

**Marks in Maths >= 65**

**Marks in Physics >= 55**

**Marks in Chemistry >= 50**

**Or**

**Total in all three subjects >= 180**

**Sample Test Cases**

**Test Case 1**

**Input**

**70 60 80**

**Output**

**The candidate is eligible**

**Test Case 2**

**Input**

**50 80 80**

**Output**

**The candidate is eligible**

**Test Case 3**

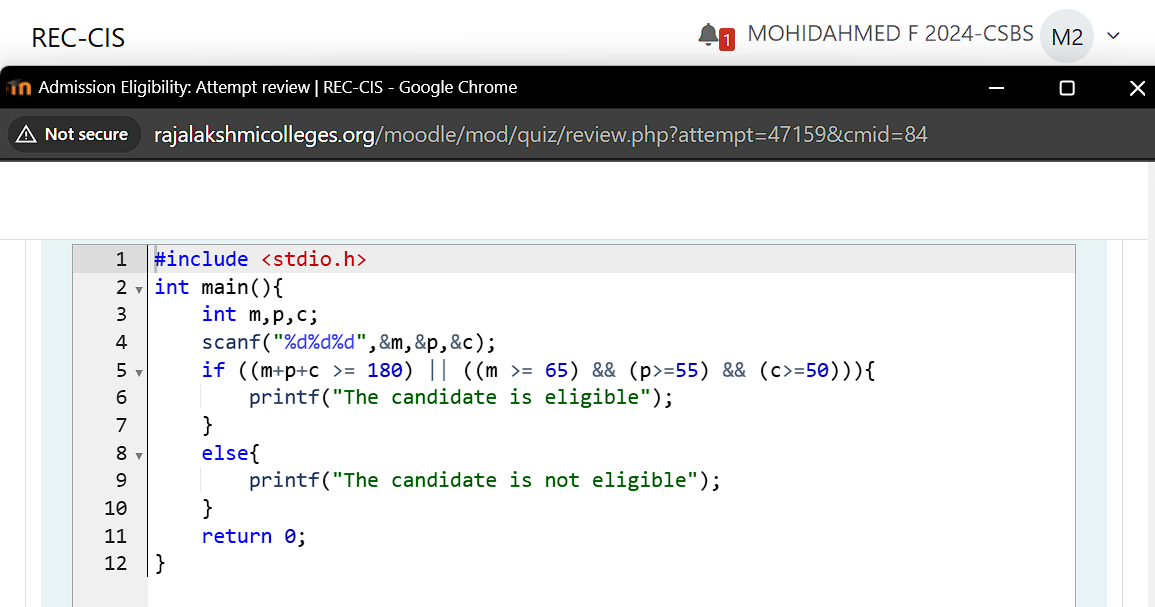
**Input**

**50 60 40**

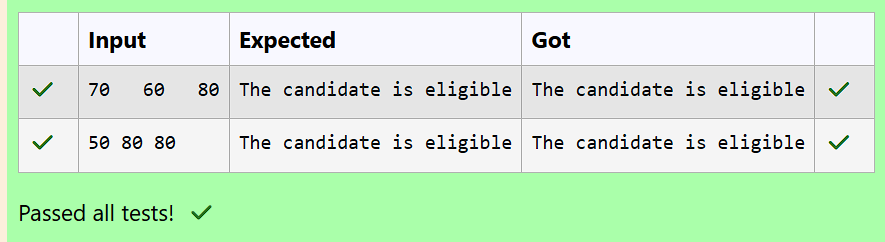
**Output**

**The candidate is not eligible**

**Program:**

****

**Output:**

****

**Question 2:**

**Complete the calculator program with Basic operations (+, -, \*, /, %) of two numbers using switch statement.**

**Sample Test Cases**

**Test Case 1**

**Input**

**45**

**45**

**+**

**Output**

**Result: 45 + 45 = 90.000000**

**Test Case 2**

**Input**

**56**

**8**

**%**

**Output**

**Result: 56 % 8 = 0.000000**

**Test Case 3**

**Input**

**50**

**70**

**$**

**Output**

**Invalid operation.**

**Result: 50 $ 70 = 0.000000**

**Test Case 4**

**Input**

**5**

**2**

**/**

**Output**

**Result: 5 / 2 = 2.500000**

**For example:**

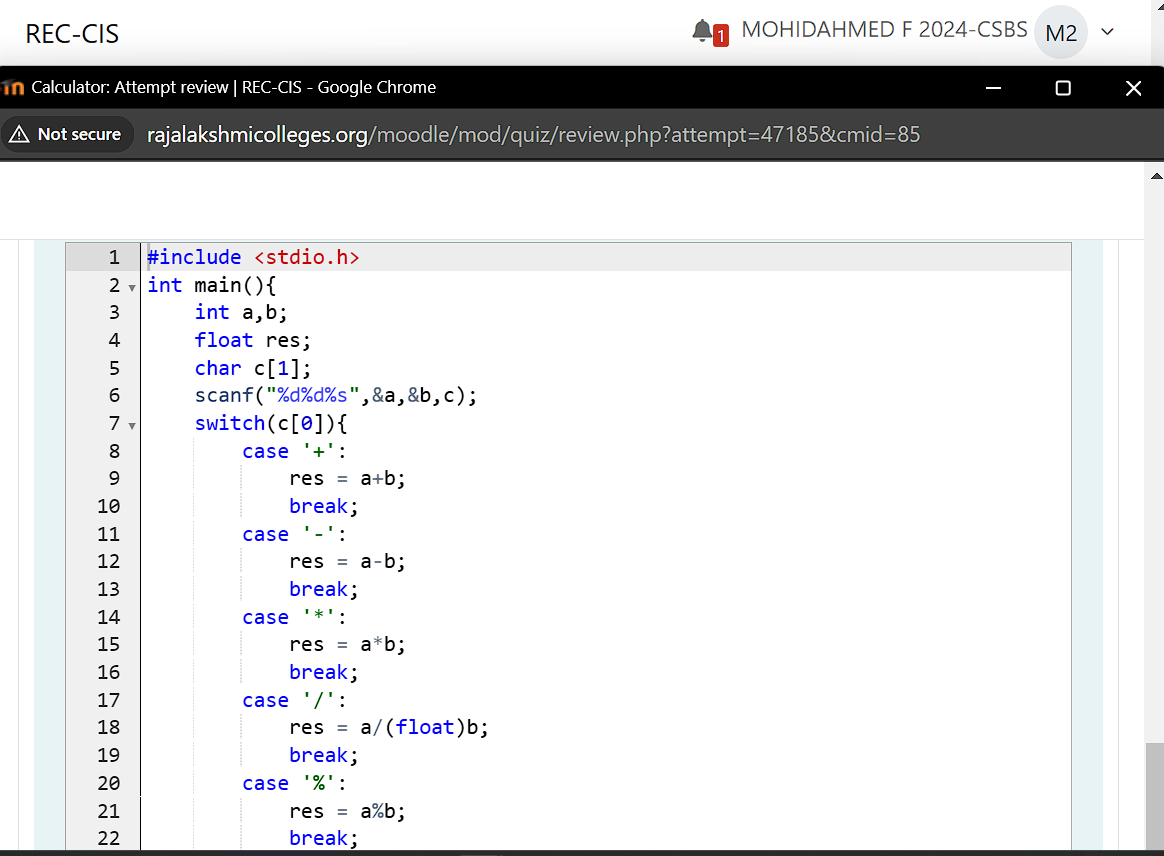
**|Input| Result|**

**|5| Result: 5 / 2 = 2.500000|**

**|2|**

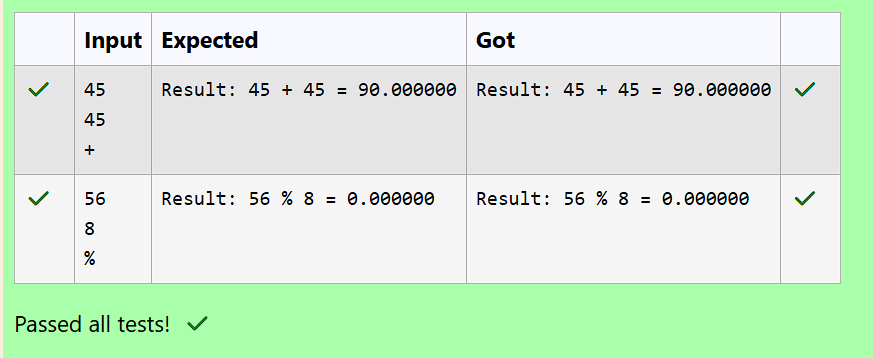
**|/|**

**Program:**

****

****

**Output:**

****

**Question 3:**

**You are given a sequence of integers as input, terminated by a -1. (That is, the input integers may be positive, negative or 0. A -1 in the input signals the end of the input.)**

**-1 is not considered as part of the input.**

**Find the second largest number in the input. You may not use arrays.**

**Sample Test Cases**

**Test Case 1**

**Input**

**-840 -288 -261 -337 -335 488 -1**

**Output**

**-261**

**Test Case 2**

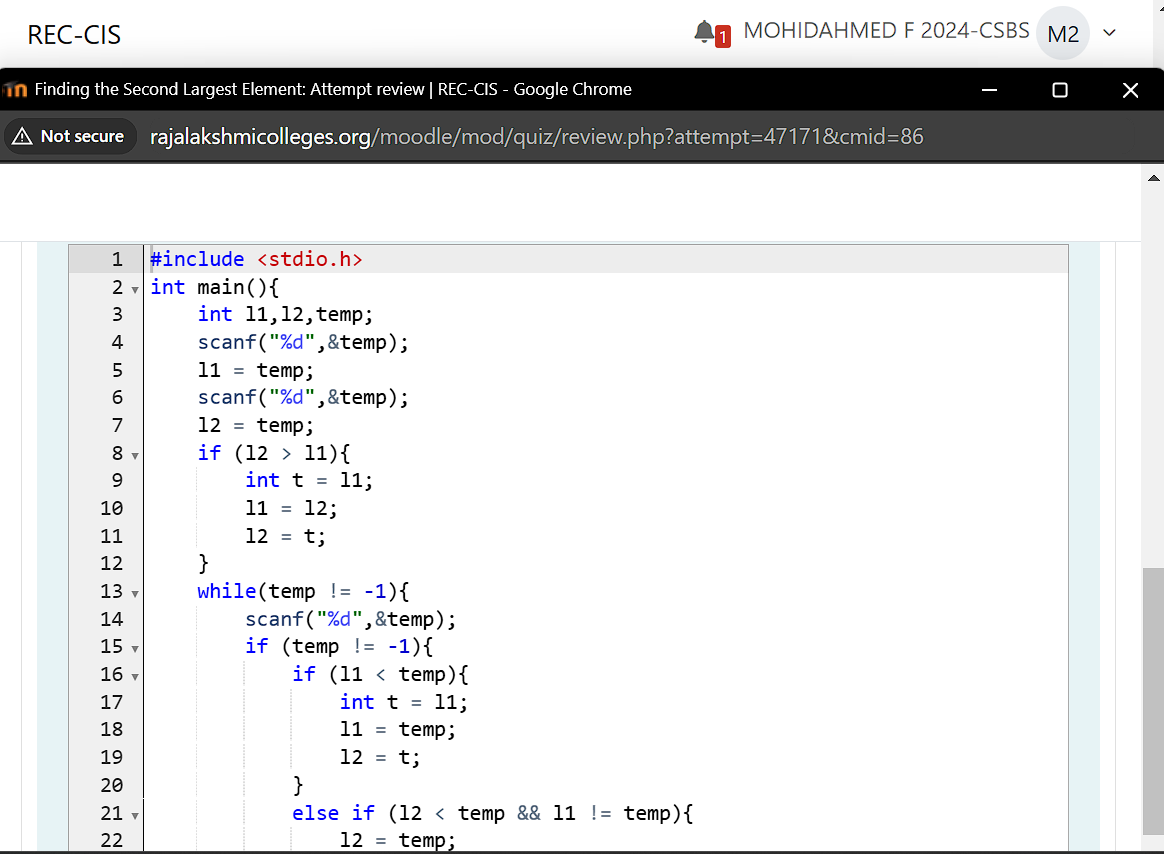
**Input**

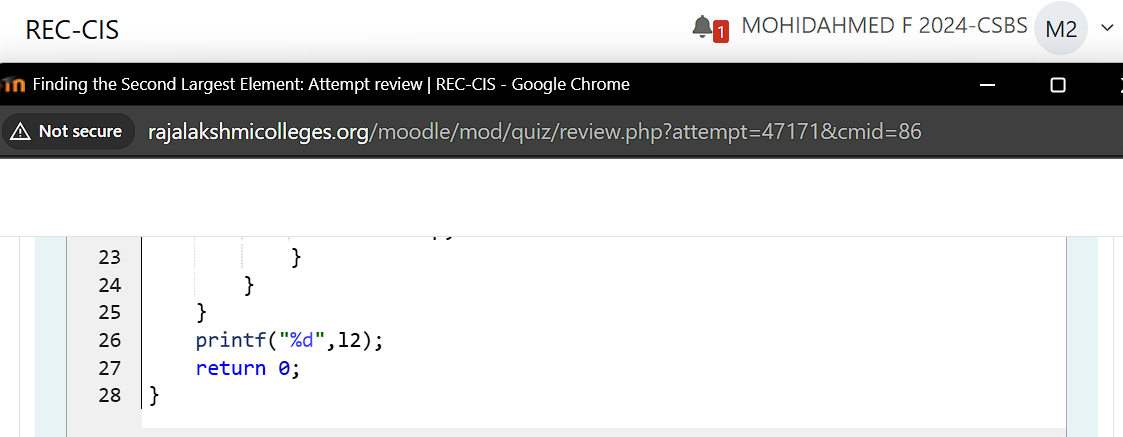
**-840 -335 -1**

**Output**

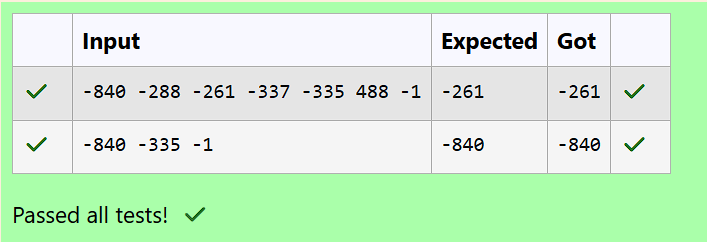
**-335**

**Program:**

****

****

**Output:**

****

**Question 4:**

**The lengths of the sides of a triangle X, Y and Z are passed as the input. The program must print the smallest side as the output.**

**Input Format:**

**The first line denotes the value of X.**

**The second line denotes the value of Y.**

**The third line denotes the value of Z.**

**Output Format:**

**The first line contains the length of the smallest side.**

**Boundary Conditions:**

**1 <= X <= 999999**

**1 <= Y <= 999999**

**1 <= Z <= 999999**

**Example Input/Output 1:**

**Input:**

**40**

**30**

**50**

**Output:**

**30**

**Example Input/Output 2:**

**Input:**

**15**

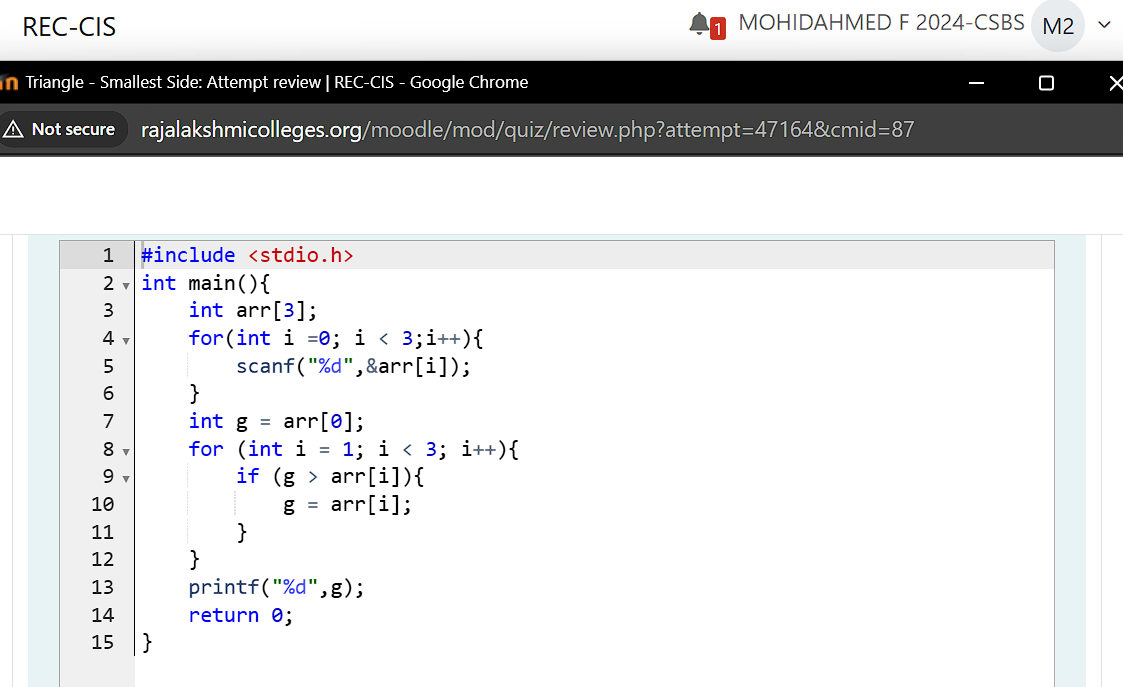
**15**

**15**

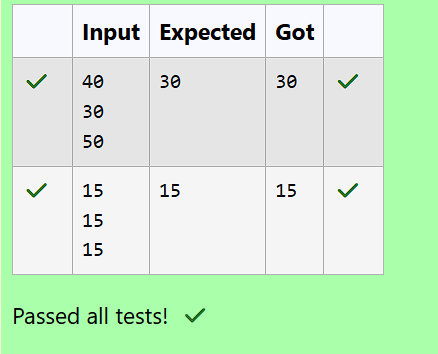
**Output:**

**15**

**Program:**

****

**Output:**

****

**Question 5:**

**An argument is an expression which is passed to a function by its caller in order for the function to perform its task. It is an expression in the comma-separated list bound by the parentheses in a function call expression.**

**A function may be called by the portion of the program with some arguments and these arguments are known as actual arguments (or) original arguments.**

**Actual arguments are local to the particular function. These variables are placed in the function declaration and function call. These arguments are defined in the calling function.**

**The parameters are variables defined in the function to receive the arguments.**

**Formal parameters are those parameters which are present in the function definition.**

**Formal parameters are available only with in the specified function. Formal parameters belong to the called function.**

**Formal parameters are also the local variables to the function. So, the formal parameters are occupied memory when the function execution starts and they are destroyed when the function execution completed.**

**Let us consider the below example:**

**#include <stdio.h>**

**int add(int, int);**

**int main()**

**{**

**int a = 10, b = 20;**

**printf("Sum of two numbers = %d\n", add(a, b)); // variables a, b are called actual arguments**

**return 0;**

**}**

**int add(int x, int y)**

**{**

**// variables x, y are called formal parameters**

**return (x + y);**

**}**

**C**

**#include <stdio.h>**

**int add(int, int);**

**int main()**

**{**

**int a = 10, b = 20;**

**printf("Sum of two numbers = %d\n", add(a, b)); // variables a, b are called actual arguments**

**return 0;**

**}**

**int add(int x, int y)**

**{**

**// variables x, y are called formal parameters**

**return (x + y);**

**}**

**In the above code whenever the function call add(a, b) is made, the execution control is transferred to the function definition of add().**

**The values of actual arguments a and b are copied into the formal arguments x and y respectively.**

**The formal parameters x and y are available only within the function definition of add(). After completion of execution of add(), the control is transferred back to the main().**

**See & retype the below code which will demonstrate about formal and actual arguments.**

**#include <stdio.h>**

**int sum(int);**

**int main()**

**{**

**int number;**

**scanf("%d", &number);**

**printf("Sum of %d natural numbers = %d\n", number, sum(number));**

**return 0;**

**}**

**int sum(int value)**

**{**

**int i, total = 0;**

**for (i = 1; i <= value; i++)**

**{**

**total = total + i;**

**}**

**return(total);**

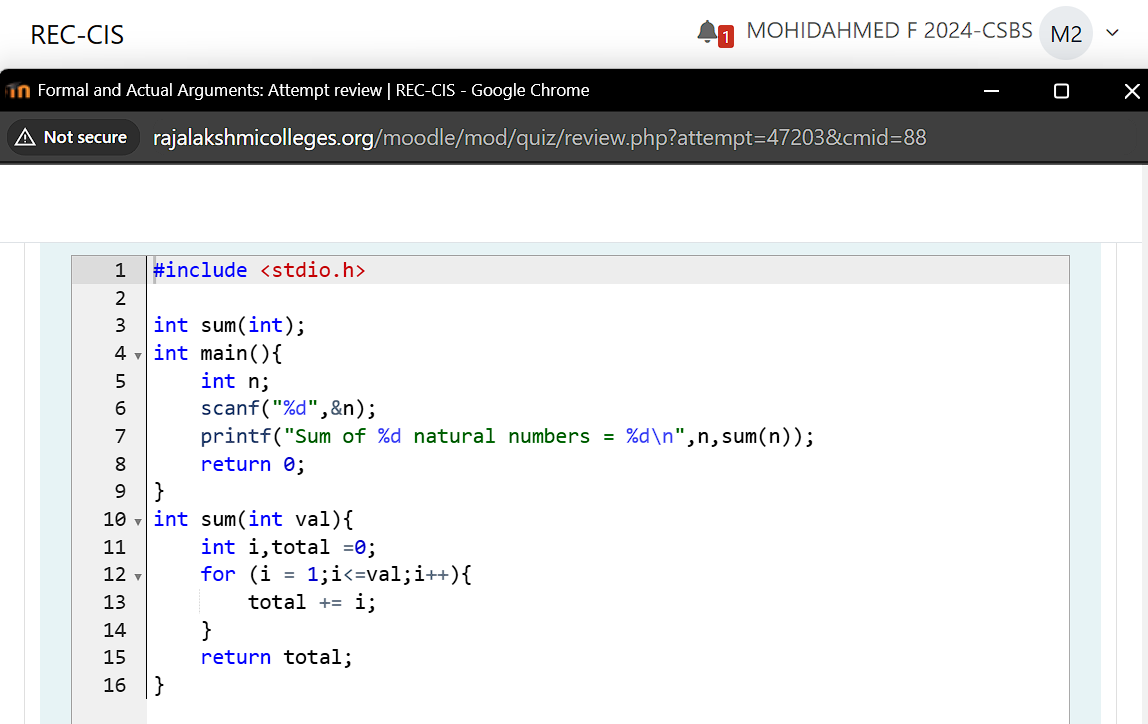
**}**

**For example:**

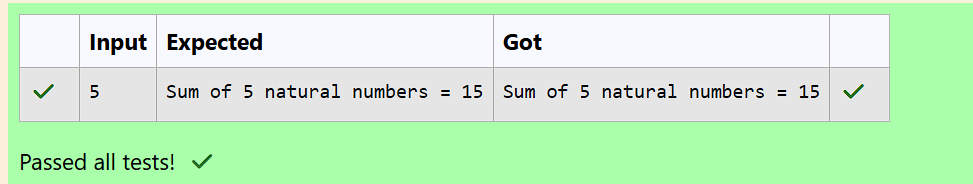
**|Input| Result|**

**|5| Sum of 5 natural numbers = 15|**

**Program:**

****

**Output:**

****

**Question 5:**

**Global variables are declared outside of any function.**

**A global variable is visible to any every function and can be used by any piece of code.**

**Unlike local variable, global variables retain their values between function calls and throughout the program execution.**

**Let us consider an example:**

**#include <stdio.h>**

**int a = 20; // Global declaration**

**void test();**

**int main()**

**{**

**printf("In main() function a = %d\n", a); // Prints 20**

**test();**

**a = a + 15; // Uses global variable**

**printf("In main() function a = %d\n", a); // Prints 55**

**return 0;**

**}**

**void test()**

**{**

**a = a + 20; // Uses global variable**

**printf("In test() function a = %d\n", a); // Prints 40**

**}**

**In the above code the global variable a is declared outside of all the functions. So, the variable a can be accessed in every function.**

**Operating System calls the main() function at the time of execution. the variable a has no local declaration, so it access the global variable a.**

**In test() function also there is no local declaration of variable a, the variable a gets access from the global.**

**The global variables are destroyed only after completion of execution of entire program.**

**See & retype the below code which will demonstrate about global variables.**

**#include <stdio.h>**

**int a = 20;**

**void test();**

**int main()**

**{**

**printf("In main() function a = %d\n", a);**

**test();**

**a = a + 15;**

**printf("In main() function a = %d\n", a);**

**return 0; //Important to add return 0; to main**

**}**

**void test() { //Also important to add a function body for test()**

**a = a + 20;**

**printf("In test() function a = %d\n", a);**

**}**

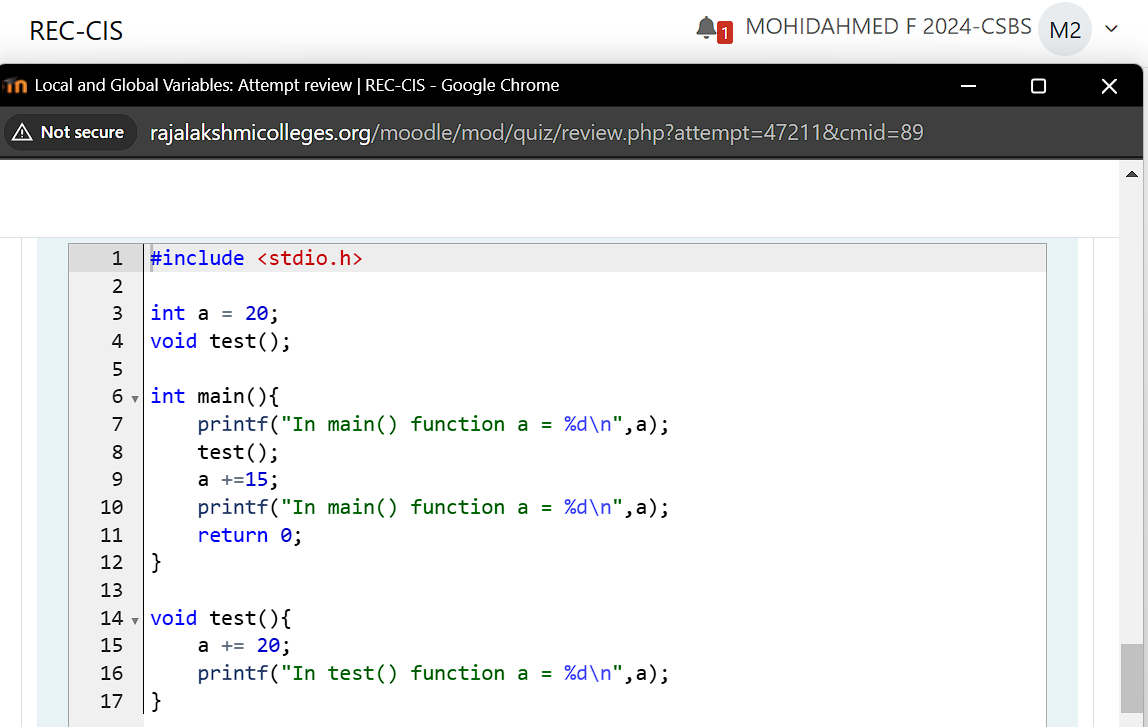
**Result**

**In main() function a = 20**

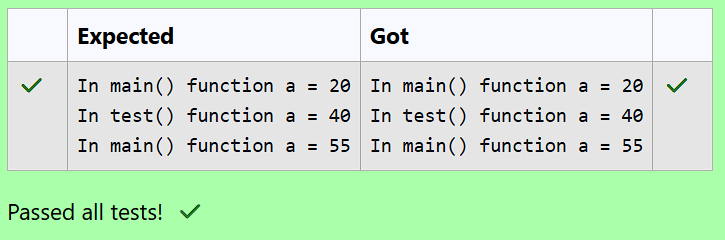
**In test() function a = 40**

**In main() function a = 55**

**Program:**

****

**Output:**

****

**Question 6:**

**Local variables are declared and used inside a function (or) in a block of statements.**

**Local variables are created at the time of function call and destroyed when the function execution is completed.**

**Local variables are accessible only with in the particular function where those variables are declared.**

**Global variables are declared outside of all the function blocks and these variables can be used in all functions.**

**Global variables are created at the time of program beginning and reside until the end of the entire program.**

**Global variables are accessible in the entire program.**

**If a local and global variable have the same name, then local variable has the highest precedence to access with in the function.**

**Let us consider an example:**

**#include <stdio.h>**

**void change();**

**int x = 20; // Global Variable x**

**int main()**

**{**

**int x = 10; // Local Variable x**

**change();**

**printf("%d", x); // The value 10 is printed**

**return 0;**

**}**

**void change()**

**{**

**printf("%d", x); // The value 20 is printed**

**}**

**In the above code the global and local variables have the same variable name x, but they are different.**

**In main() function the local variable x is only accessed, so it prints the value 10.**

**In change() function the variable x is not declared locally so it access global variable x, so it prints 20.**

**See & retype the below code which will demonstrate about local and global variables**

**#include <stdio.h>**

**int x = 15;**

**void change1(int x)**

**{**

**printf("In change1() function x = %d\n", x);**

**}**

**void change2()**

**{**

**printf("In change2() function x = %d\n", x);**

**}**

**int main()**

**{**

**int x = 10;**

**printf("In main() function x = %d\n", x);**

**change1(x);**

**change2();**

**printf("In main() function x = %d\n", x);**

**return 0;**

**}**

**For example:**

**Result:**

**In main() function x = 10**

**In change1() function x = 10**

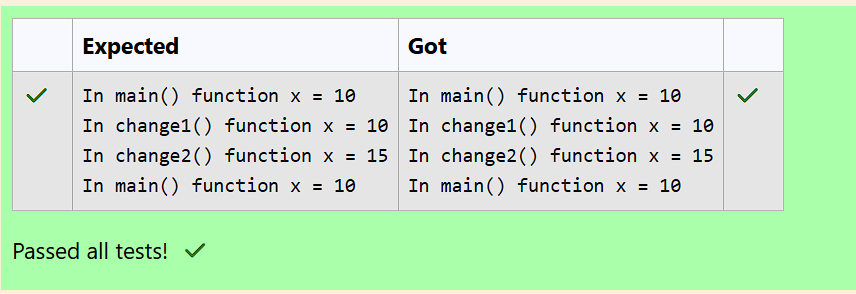
**In change2() function x = 15**

**In main() function x = 10**

**Program:**

****

**Output:**

****

**Question 7:**

**All the C functions can be called either with arguments or without arguments in a C program. These functions may or may not return values to the calling function.**

**Depending on the arguments and return values functions are classified into 4 categories.**

1. **Function without arguments and without return value**
2. **Function with arguments and without return value**
3. **Function without arguments and with return value**
4. **Function with arguments and with return value**

**When a function has no arguments, it does not receive any data from the calling function.**

**Similarly, when a function does not return a value, the calling function does not receive any data from the called function.**

**In effect, there is no data transfer between the calling function and the called function in the category function without arguments and without return value.**

**Let us consider an example of a function without arguments and without return value:**

**#include <stdio.h>**

**void india\_capital(void);**

**int main()**

**{**

**india\_capital();**

**return 0;**

**}**

**void india\_capital()**

**{**

**printf("New Delhi is the capital of India\n");**

**}**

**C**

**void india\_capital()**

**{**

**printf("New Delhi is the capital of India\n");**

**}**

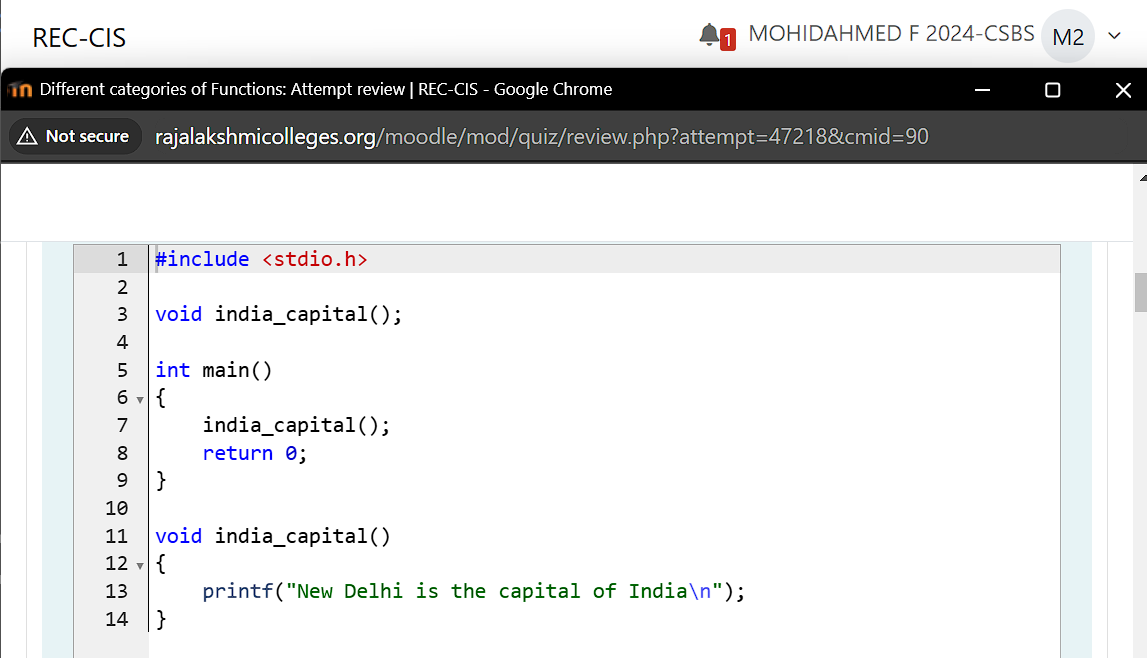
**In the above sample code the function void india\_capital(void); specifies that the function does not receive any arguments and does not return any value to the main() function.**

**Identify the below errors and correct them.**

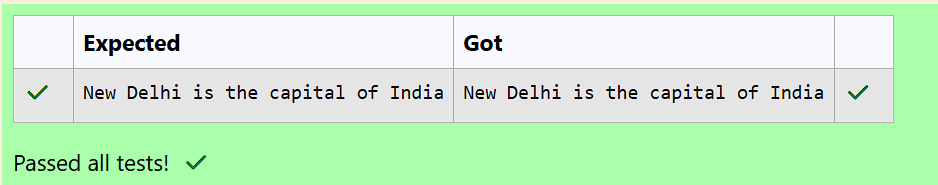
**For example:**

|  |
| --- |
| **Result:** |
| **New Delhi is the capital of India** |

**Program:**

****

**Output:**

****

**Question 8:**

**Write a C program to demonstrate functions without arguments and without return value.**

**Write the functions print() and hello().**

**The output is:**

**…\*\*\*…**

**Hello! REC**

**…\*\*\*…**

**For example:**

|  |
| --- |
| **Result** |

**…\*\*\*…**

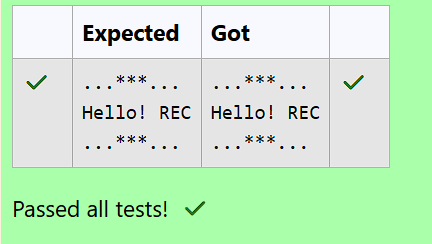
**Hello! REC**

**…\*\*\*…**

**Program:**

****

**Output:**

****

**Question 9:**

**When a function definition has arguments, it receives data from the calling function.**

**The actual arguments in the function call must correspond to the formal parameters in the function definition, i.e. the number of actual arguments must be the same as the number of formal parameters, and each actual argument must be of the same data type as its corresponding formal parameter.**

**The formal parameters must be valid variable names in the function definition and the actual arguments may be variable names, expressions or constants in the function call.**

**The variables used in actual arguments must be assigned values before the function call is made. When a function call is made, copies of the values of actual arguments are passed to the called function.**

**What occurs inside the function will have no effect on the variables used in the actual argument list. There may be several different calls to the same function from various places with a program.**

**Let us consider an example of a function with arguments and without return value**

**#include <stdio.h>**

**void largest(int, int);**

**int main()**

**{**

**int a, b;**

**printf("Enter two numbers: ");**

**scanf("%d%d", &a, &b);**

**largest(a, b);**

**return 0;**

**}**

**void largest(int x, int y)**

**{**

**if (x > y)**

**{**

**printf("Largest element = %d\n", x);**

**}**

**else**

**{**

**printf("Largest element = %d\n", y);**

**}**

**}**

**In the above sample code the function void largest(int, int); specifies that the function receives two integer arguments from the calling function and does not return any value to the called function.**

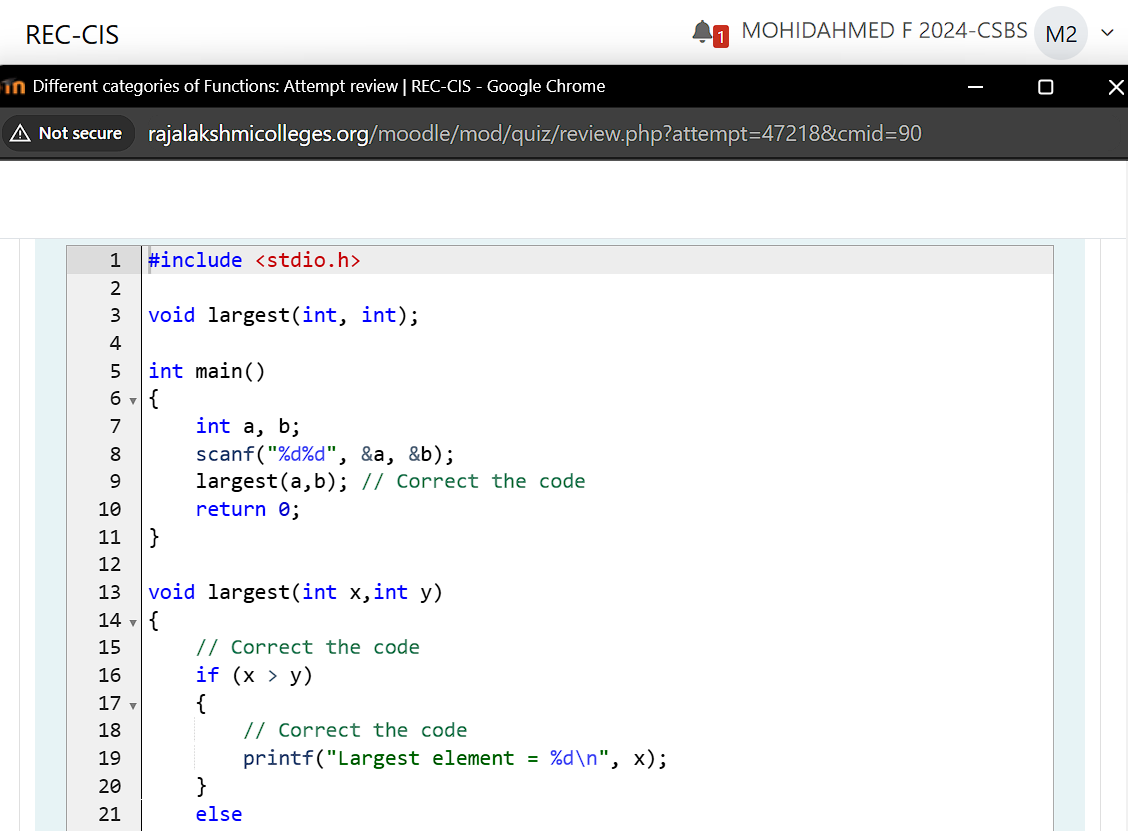
**When the function call largest(a, b) is made in the main() function, the values of actual arguments a and b are copied in to the formal parameters xand y. After completion of execution of largest(int x, int y) function, it does not return any value to the main() function. Simply the control is transferred to the main() function.**

**Fill in the missing code in the below program to find the largest of two numbers using largest() function**

**For example:**

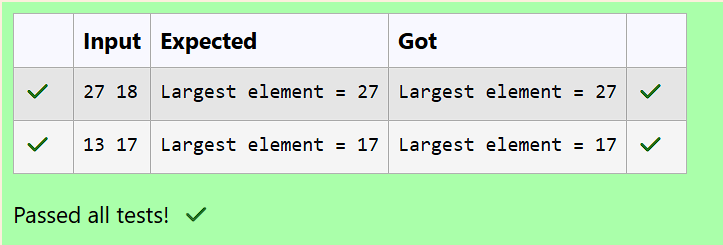
|  |  |
| --- | --- |
| **Input** | **Result** |
| **27 18** | **Largest element = 27** |
| **13 17** | **Largest element = 17** |

**Program:**

****

****

**Output:**

****

**Question 10:**

**Fill the missing code to understand the concept of a function with arguments and without return value.**

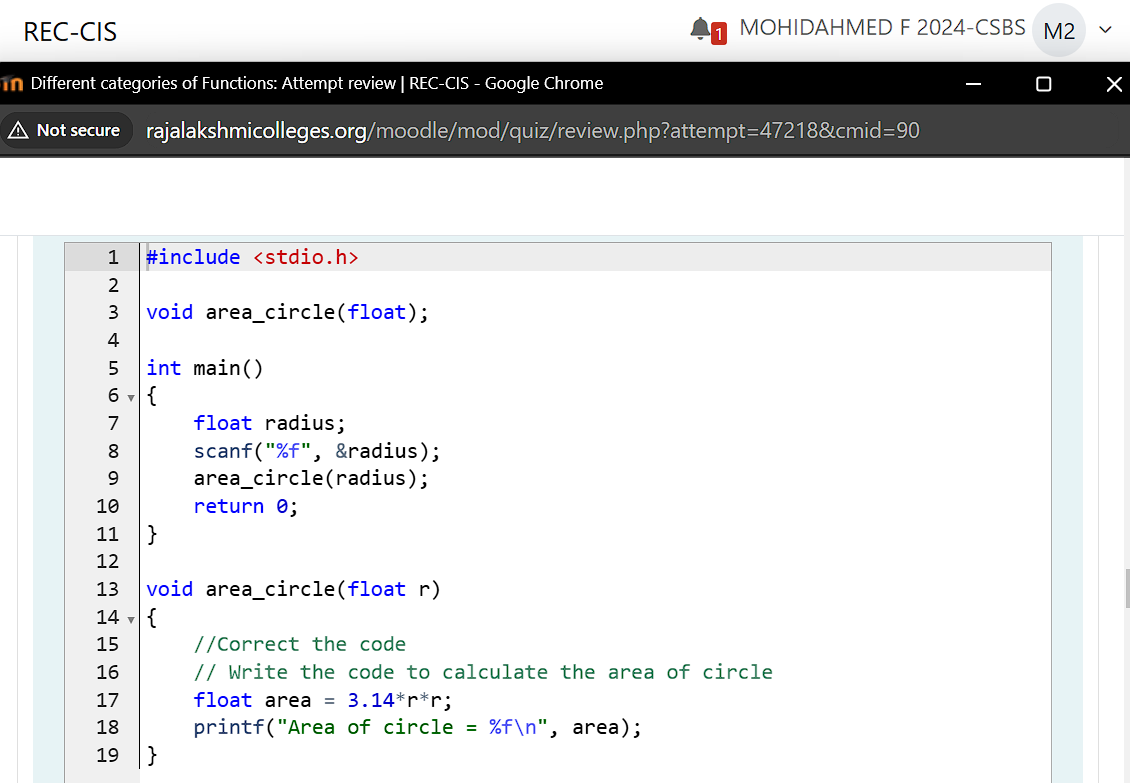
**Note: Take pi value as 3.14**

**The below code is to find the area of circle using functions.**

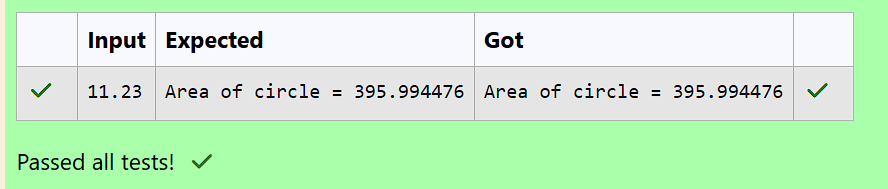
**For example:**

|  |  |
| --- | --- |
| **|Input|** | **Result |** |
| **|11.23|** | **Area of circle = 395.994476** |

**Program:**

****

**Output:**

****

**Question 11:**

**When a function has no arguments, it does not receive any data from the calling function.**

**When a function return a value, the calling function receives data from the called function.**

**Let us consider an example of a function without arguments and with return value:**

**#include <stdio.h>**

**int sum(void);**

**int main()**

**{**

**printf("\nSum of two given values = %d\n", sum());**

**return 0;**

**}**

**int sum()**

**{**

**int a, b, total;**

**printf("Enter two numbers: ");**

**scanf("%d%d", &a, &b);**

**total = a + b;**

**return total;**

**}**

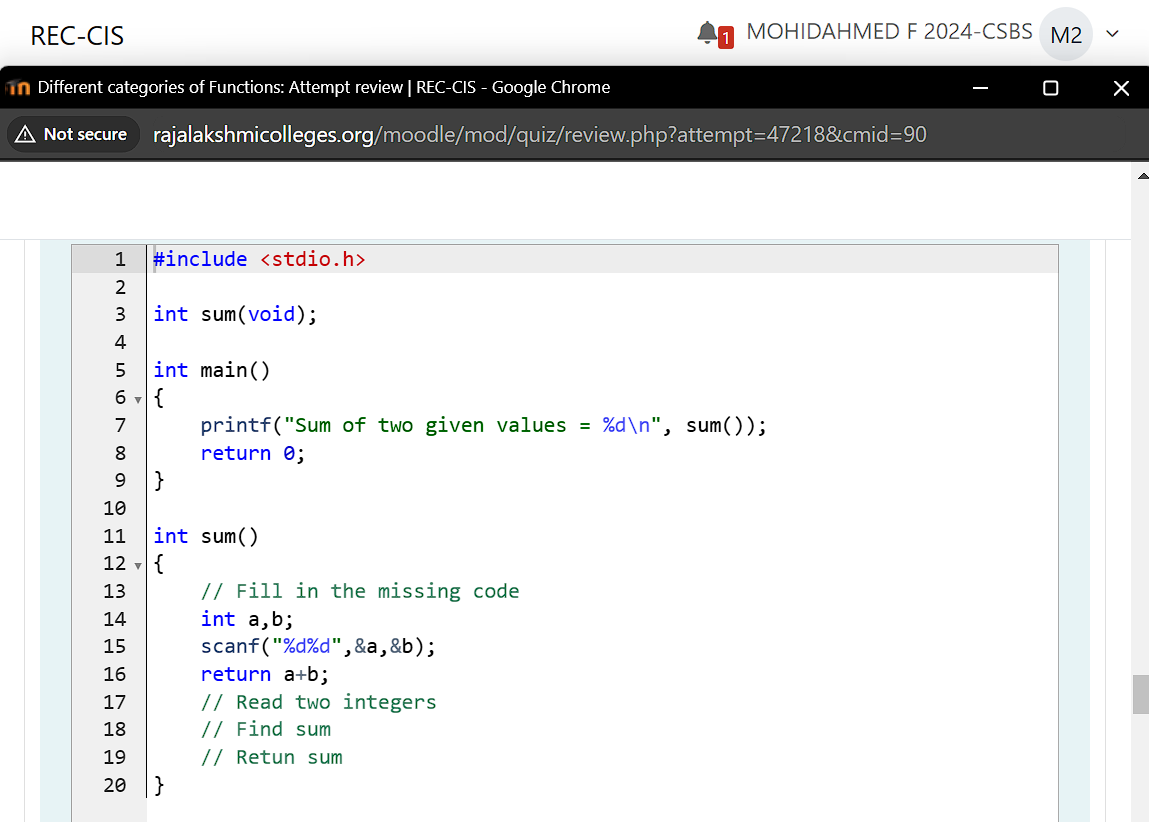
**In the above sample code the function int sum(void); specifies that the function does not receive any arguments but return a value to the calling function.**

**Fill in the missing code in the below program to find sum of two integers.**

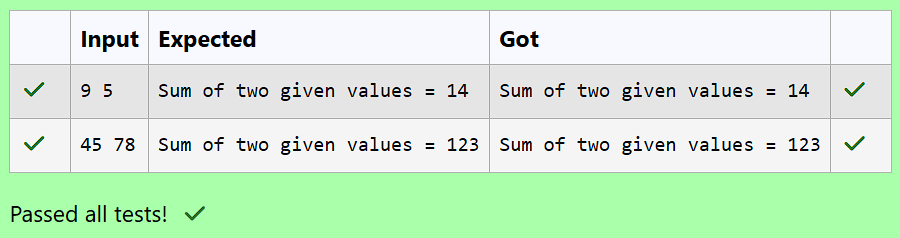
**For example:**

|  |  |
| --- | --- |
| **Input** | **Result** |
| **9 5** | **Sum of two given values = 14** |
| **45 78** | **Sum of two given values = 123** |
|  |  |

**Program:**

****

**Output:**

****

**Question 12:**

**When a function definition has arguments, it receives data from the calling function.**

**After taking some desired action, only one value will be returned from called function to calling function through the return statement.**

**If a function returns a value, the function call may appear in any expression and the returned value used as an operand in the evaluation of the expression.**

**Let us consider an example of a function with arguments and with return value:**

**#include <stdio.h>**

**int largest(int, int, int);**

**int main()**

**{**

**int a, b, c;**

**printf("Enter three numbers: ");**

**scanf("%d%d%d", &a, &b, &c);**

**printf(" Largest of the given three numbers = %d\n", largest(a, b, c));**

**return 0;**

**}**

**int largest(int x, int y, int z)**

**{**

**if ((x > y) && (x > z))**

**{**

**return x;**

**}**

**else if (y > z)**

**{**

**return y;**

**}**

**else**

**{**

**return z;**

**}**

**}**

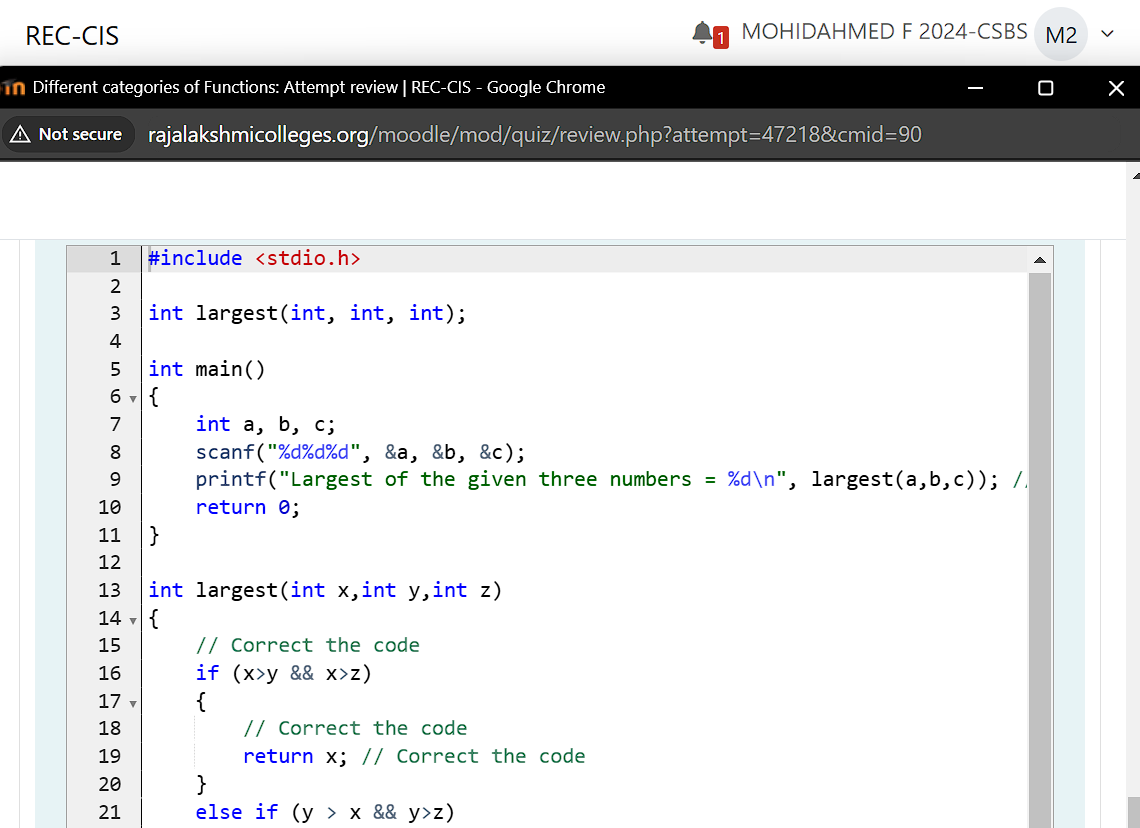
**In the above sample code the function int largest(int, int, int); specifies that the function receives three values and returns a value to the calling function.**

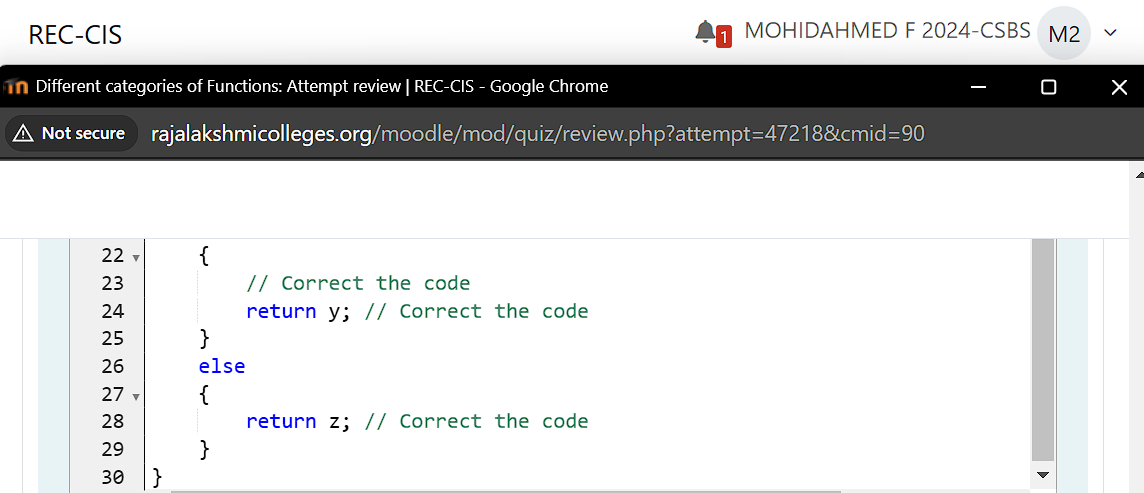
**Fill in the missing code in the below program to find the largest of three numbers using largest() function.**

**For example:**

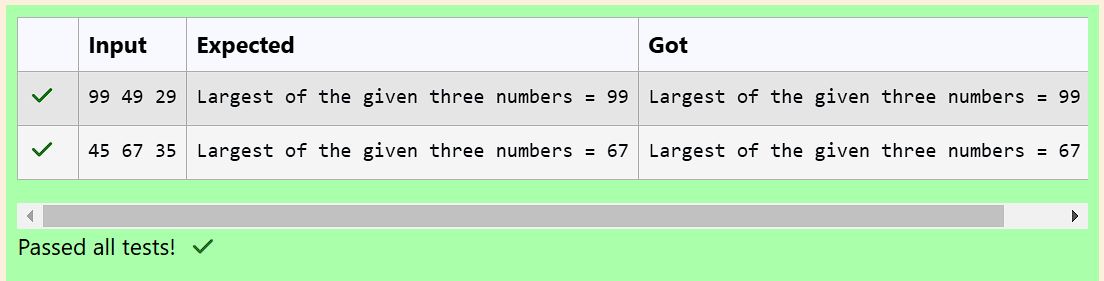
|  |  |
| --- | --- |
| **Input** | **Result** |
| **99 49 29** | **Largest of the given three numbers = 99** |
| **45 67 35** | **Largest of the given three numbers = 67** |

**Program:**

****

****

**Output:**

****

**Question 13:**

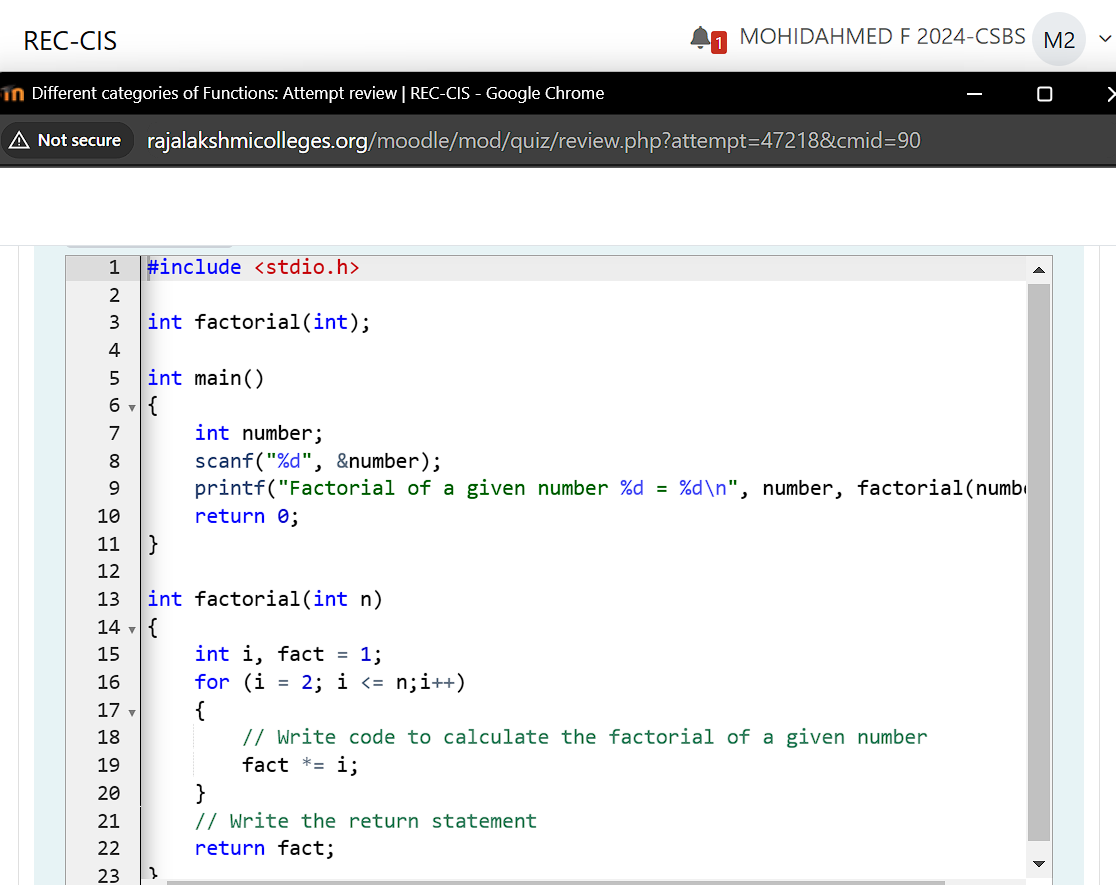
**Fill in the missing code in the below code to understand about function with arguments and with return value.**

**The below code is to find the factorial of a given number using functions.**

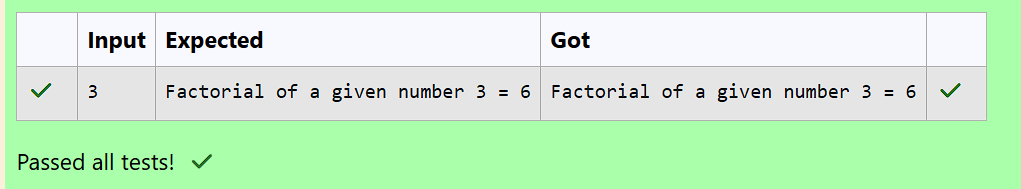
**For example:**

|  |  |
| --- | --- |
| **Input** | **Result** |
| **3** | **Factorial of a given number 3 = 6** |

**Program:**

****

**Output:**

****

**Question 14:**

**Write a C program to demonstrate functions without arguments and with return value.**

**The below code is used to check whether the given number is a prime number or not.**

**Write the function prime().**

**Sample Input and Output:**

**5**

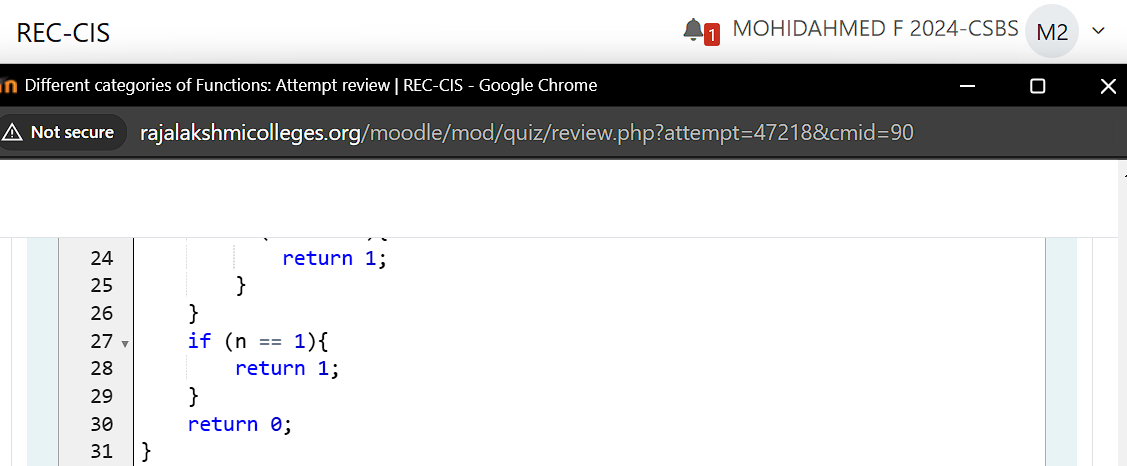
**The given number is a prime number**

**For example:**

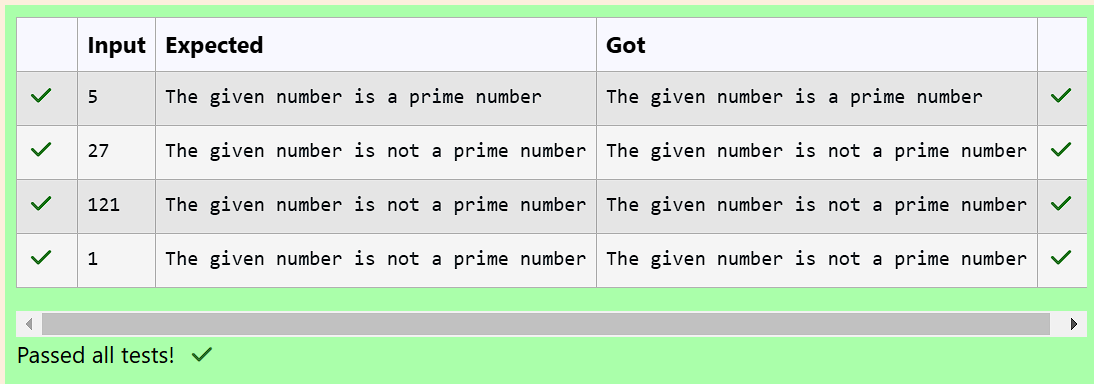
|  |  |
| --- | --- |
| **Input** | **Result** |
| **5** | **The given number is a prime number** |
| **27** | **The given number is not a prime number** |
| **121** | **The given number is not a prime number** |
| **1** | **The given number is not a prime number** |

**Program:**

****

****

**Output:**

****