Question 1	Write a C program to find the eligibility of admission for a professional course based on the following criteria:
Correct	
Marked out of	Marks in Maths >= 65
1.00	Marks in Physics >= 55
P Flag question	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
	The Candidate is not eligible

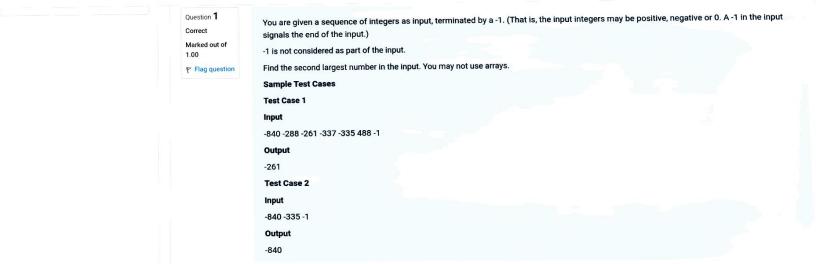
REU-UIS

Answer: (penalty regime: 0 %)

```
1 |#include<stdio.h>
 2 int main()
 3 + {
        int math,physics,chem,total;
       scanf("%d %d %d",&math,&physics,&chem);
 5
        total=math+physics+chem;
 6
        if((math>=65 && physics>=55 && chem>=50)||(total>=180))
 7
 8 .
           printf("The candidate is eligible");
 9
10
       else
11
12 +
           printf("The candidate is not eligible");
13
14
       return 0;
15
16 }
```

	Input	Expected	Got	
~	70 60 80	The candidate is eligible	The candidate is eligible	~
~	50 80 80	The candidate is eligible	The candidate is eligible	~

Finish review



```
#include<stdio.h>
    #include<limits.h>
    int main()
4 . {
        int largest=INT_MIN,second_largest=INT_MIN;
        while(1)
 8 .
            scanf("%d",&n);
10
            if(n ==-1)
11 .
12
                break;
13
14
            if(n>largest)
15 +
16
                second_largest=largest;
17
                largest=n;
18
19
            else if(n>second_largest&& n!=largest)
20 •
21
            second_largest=n;
22
23
24
        if(second_largest !=INT_MIN)
25 •
26
            printf("%d",second_largest);
27
28
29 }
```

Passed a

out					Expected	Got	
40 -288	-261 -33	37 -335	488	-1	-261	-261	~

1000				
li	nput	Expected	Got	
-	840 -288 -261 -337 -335 488 -1	-261	-261	~
	840 -335 -1	-840	-840	~

Question 1 Correct	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.
Marked out of 1.00	Input Format:
₹ Flag question	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
	Answer: (penalty regime: 0 %)

REC-CIS

```
scanf("%d %d %d",&X,&Y,&Z);
int smallest=X;
6
         if(Y<smallest)</pre>
7
8 .
              smallest=Y;
10
11
         if(Z<smallest)</pre>
12 +
13
              smallest=Z;
14
15
         printf("%d\n", smallest);
16
17 }
```

|#include<stdio.h> int main()

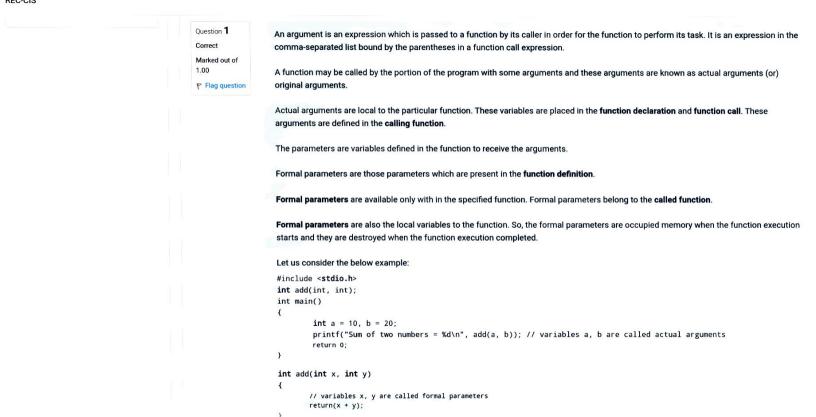
int X,Y,Z;

2 3 . {

	Input	Expected	Got	
~	40 30 50	30	30	~
~	15 15 15	15	15	~

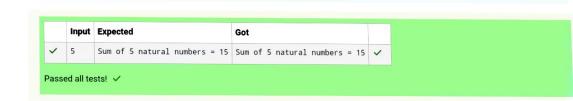
Passed all tests! 🗸

Finish review



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 in to the formal arguments x and y respectively. The formal parameters x and y are available only with in 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 Sum of 5 natural numbers = 15

```
5
      Sum of 5 natural numbers = 15
Answer: (penalty regime: 0 %)
      #include<stdio.h>
      int sum(int);
   3 int main()
   4 - {
          int number;
          scanf ("%d",&number);
          printf("Sum of %d natural numbers = %d\n",number,sum(number));
   8
          return 0:
   9
      int sum(int value)
  11 + {
  12
          int i,total=0;
  13
          for(i=1;i<=value;i++)</pre>
  14 .
  15
               total=total+i;
  16
  17
  18
           return(total);
  19 }
```



Ouestion 1 Incorrect Marked out of 1.00 Flag question

A local variable is declared inside a function.

Let us consider an example:

A local variable is visible only inside their function, only statements inside function can access that local variable.

#include <stdio.h>
void test();
int main()
{
 int a = 22, b = 44;
 test();
 printf("Values in main() function a = %d and b = %d\n", a, b);
 return 0;
}

void test()
{
 int a = 50, b = 80;
 printf("Values in test() function a = %d and b = %d\n", a, b);

Local variables are declared when the function execution started and local variables gets destroyed when control exits from function.

In the above code we have 2 functions main() and test(), in these functions local variables are declared with same variable names a and b but they are different.

Operating System calls the main() function at the time of execution. the local variables with in the main() are created when the main() starts execution.

After completion of execution of test() function, the local variables are destroyed and the control is transferred back to the main() function.

Operating System calls the main() function at the time of execution. the local variables with in the main() are created when the main() starts execution.

when a call is made to test() function, first the control is transferred from main() to test(), next the local variables with in the test() are created and they are available only with in the test() function.

```
when a call is made to test() function, first the control is transferred from main() to test(), next the local variables with in the test() are created
and they are available only with in the test() function.
After completion of execution of test() function, the local variables are destroyed and the control is transferred back to the main() function.
See & retype the below code which will demonstrate about local variables.
#include <stdio.h>
void test():
int main()
  int a = 9. b = 99:
  test():
   printf("Values in main() function a = %d and b = %d\n", a, b);
  return 0;
```

void test() int a = 5, b = 55;

execution.

printf("Values in test() function a = %d and b = %d\n", a, b);

Values in main() function a = 9 and b = 99

Operating System calls the main() function at the time of execution, the local variables with in the main() are created when the main() starts

For example:

Result Values in test() function a = 5 and b = 55 REC-CIS

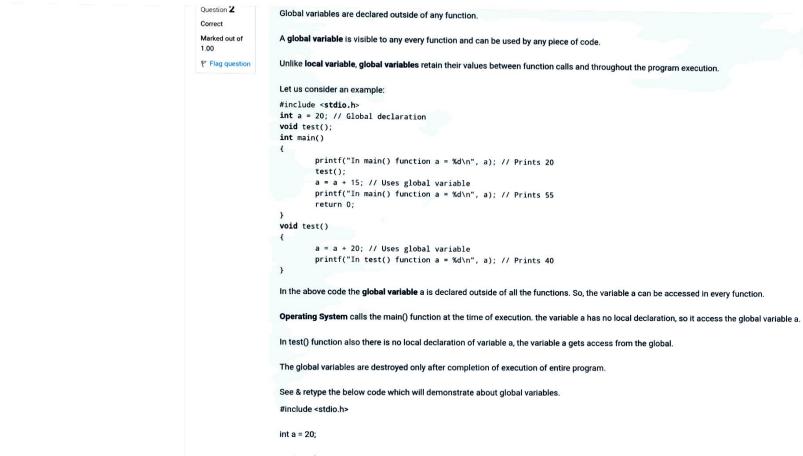
```
Answer: (penalty regime: 0 %)
     |#include<stdio.h>
   2 void test();
      int main()
   4 - {
         int a = 9, b = 99;
   5
          test();
          printf("Values in main() function a = %d and b = %d\n",a , b);
   8 return 0:
  10
     void test()
  11 - {
  12
          int a = 5, b = 55;
  13
          printf("Values in test() function a = %d and b = %d\n ",a,b);
  14 }
```

```
Expected

X Values in test() function a = 5 and b = 55 Values in main() function a = 9 and b = 99

Your code must pass all tests to earn any marks. Try again.

Show differences
```



```
int main()
  printf("In main() function a = %d\n", a);
  test();
  a = a + 15;
  printf("In main() function a = %d\n", a);
  return 0;
 void test()
   a = a + 20;
   printf("In test() function a = %d\n", a);
 For example:
   Result
   In main() function a = 20
   In test() function a = 40
```

In main() function a = 55

```
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);
10
       return 0;
11 }
12
   void test()
13 - {
14
       a = a + 20;
15
       printf("In test() function a = %d\n",a);
16 }
```

Passed all tests! ✓

#include<stdio.h>

Question 3

Incorrect

Local variables are created at the time of function call and destroyed when the function execution is completed. Marked out of 1.00

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

Local variables are accessible only with in the particular function where those variables are declared. F Flag question 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
```

printf("%d", x); // The value 10 is printed return 0; void change()

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

change();

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.

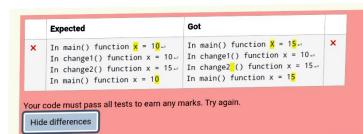
In change() function the variable x is not declared locally so it access global variable x, so it prints 20.

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
```

```
REC-CIS
```

```
1 |#include<stdio.h>
 2 int X = 15:
 3 void change1(int X)
 4 - {
        printf("In change1() function x = %d\n", X);
    void change2()
 7
 8 - {
        printf("In change2 () function x = %d\n",X);
 9
10 }
11 int main()
12 - {
        int x = 10;
13
        printf("In main() function X = %d\n",X);
14
15
        change1(x);
        change2();
16
       printf("In main() function x = %d\n",X);
17
18
        return 0;
19 }
```



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

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"); In the above sample code the function void india_capital(void); specifies that the function does not receive any arguments and does not

Identify the below errors and correct them. For example:

return any value to the main() function.

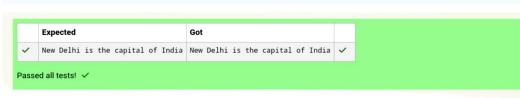
Result

Reset answer

New Delhi is the capital of India

Answer: (penalty regime: 0 %)

New Delhi is the capital of India Answer: (penalty regime: 0 %) Reset answer 1 #include <stdio.h> void india_capital(void); int main() 6 + { india_capital(); 8 return 0; 9 10 void india capital() 12 - { 13 printf("New Delhi is the capital of India\n"); 14 }



Question 2 Write a C program to demonstrate functions without arguments and without return value. Correct Write the functions print() and hello(). Marked out of 1.00 Flag question The output is: ...***... Hello! REC ...***... For example: Result ...***... Hello! REC ...***... Answer: (penalty regime: 0 %) Reset answer |#include <stdio.h> // Write the functions void hello(void) 5 + { printf("Hello! REC\n"); 6 7 9 int main() 10 + { printf("...***...\n"); 11 12 hello(); printf("...***..."); 13 14 return 0; 15 }

	Expected	Got	
~	***	***	~
	Hello! REC		
	***	***	



Passed all tests! 🗸

Correct Marked out of 1.00 P Flag question

Question 3

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

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

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

corresponding formal parameter.

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.

values of actual arguments are passed to the called function.

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);

```
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 x and v.

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:

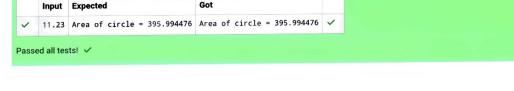
Input Result	
27 18 Largest element = 27	
13 17 Largest element = 17	

```
Answer: (penalty regime: 0 %)
 Reset answer
   1 |#include <stdio.h>
   3 void largest(int, int);
   5 int main()
   6 . {
          int a, b;
   7
          scanf("%d%d", &a, &b);
         largest(a,b); // Correct the code
  10
          return 0;
  11 }
  12
 13 void largest(int x, int y)
  14 - {
  15
          // Correct the code
  16
          if (x>y)
 17 -
             // Correct the code
  18
             printf("Largest element = %d\n", x);
 19
 20
 21
         else
 22 -
             printf("Largest element = %d\n", y);
 23
 24
 25 }
```

Input	Expected	Got	
27 18	Largest element = 27	Largest element = 27	~
13 17	Largest element = 17	Largest element = 17	~

Passed

```
Question 4
                  Fill the missing code to understand the concept of a function with arguments and without return value.
Correct
Marked out of
                  Note: Take pi value as 3.14
1.00
                  The below code is to find the area of circle using functions.
F Flag question
                  For example:
                   Input Result
                   11.23 Area of circle = 395.994476
                  Answer: (penalty regime: 0 %)
                    Reset answer
                         |#include <stdio.h>
                         #define pi 3.14
                          void area_circle(float);
                          int main()
                              float radius:
                              scanf("%f", &radius);
                      9
                              area_circle(radius);
                      10
                              return 0;
                      12
                      13
                          void area_circle(float radius)
                      15 + {
                      16
                              //Correct the code
                      17
                              // Write the code to calculate the area of circle
                              float area=pi*radius*radius;
                              printf("Area of circle = %f\n", area);
                      20 }
```



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

Question 5 Correct Marked out of 1.00

Flag question

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

#include <stdio.h> int sum(void); int main()

int sum() {

calling function.

For example:

return 0:

int a. b. total;

printf("Enter two numbers : "); scanf("%d%d", &a, &b); total = a + b;return total;

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

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

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

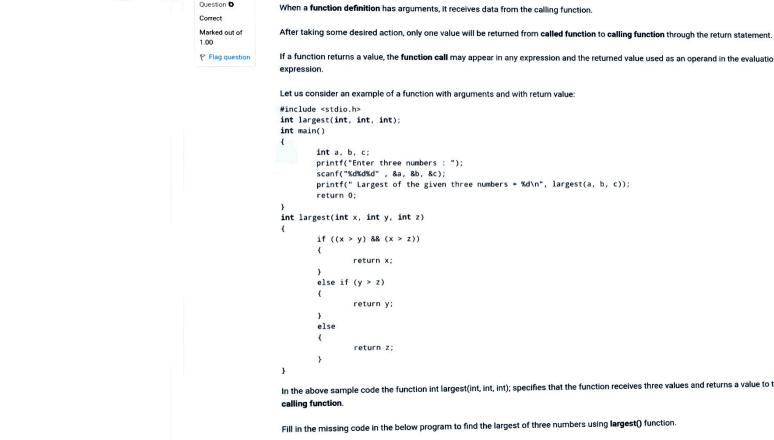
Fill in the missing code in the below program to find sum of two integers.

```
Reset answer
    #include <stdio.h>
     int sum(void);
  5
     int main()
  6 + {
         printf("Sum of two given values = %d\n", sum());
  7
  8
         return 0;
 10
11
     int sum()
 12 - {
 13
          int a,b,total;
 14
          scanf("%d %d ",&a,&b);
 15
          total=a+b;
 16
          return total;
          // Fill in the missing code
// Read two integers
// Find sum
 17
 18
 19
 20
          // Retun sum
```

21 }

REC-CIS

	Input	Expected	Got	
~	9 5	Sum of two given values = 14	Sum of two given values = 14	~
~	45 78	Sum of two given values = 123	Sum of two given values = 123	~



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);
```

return 0:

else

return z;

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

if ((x > y) && (x > z))return x: else if (v > z)return y;

} 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.

```
Input
          Result
99 49 29 Largest of the given three numbers = 99
45 67 35 Largest of the given three numbers = 67
Answer: (penalty regime: 0 %)
 Reset answer
   1 #include <stdio.h>
      int largest(int, int, int);
   5
      int main()
   6 . {
          int a, b, c;
          scanf("%d%d%d", &a, &b, &c);
          printf("Largest of the given three numbers = %d\n", largest(a,b,c)); // Correct the code
  10
          return 0;
 11 |}
 12
     int largest(int x, int y,int z)
 14 - {
 15
          // Correct the code
 16
          if((x>y)&&(x>z))
 17 -
 18
              // Correct the code
 19
              return x ; // Correct the code
 20
 21
          else if (y>z)
 22 ,
              // Correct the code
 23
 24
              return y ; // Correct the code
 25
 26
         else
 27 -
             return z; // Correct the code
 28
 29
 30 }
```

For example:

45 67 35 Largest of the given three numbers = 67 Largest of the given three numbers = 67		99 49 29	Largest of the given three numbers = 99	Largest of the given three numbers = 99	*
		45 67 35	Largest of the given three numbers = 67	Largest of the given three numbers = 67	~
	ed all tests! 🗸	~			
ests! 🗸					

Question 7
Correct
Marked out of
1.00

P Flag question

For example:
Input Result

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

3 Factorial of a given number 3 = 6

Answer: (penalty regime: 0 %)

REC-CIS

1 #	finclude <stdio.h></stdio.h>	
2		
3 i	nt factorial(int number);	
4	9	
	nt main()	
6 + {		
7	int number;	
8	SCHOOL III On what I	
9	scant("%d", &number); printf("Factorial of a given number %d = %d\n", number, factorial(number));	
10	return 0;	
11 }		
12		
13 i	nt factorial(int number)	
4 + {		
15	<pre>int i, factorial = 1;</pre>	
16	for (i=1;i<=number;i++)	
17 +		
18	// Write code to calculate the factorial of a given number	
19	<pre>factorial=factorial*i;</pre>	
0	}	
1	// Write the return statement	
22	return factorial;	
	Tetal Hutter 2027	
3 }		

	Input	Expected	Got	
~	3	Factorial of a given number 3 = 6	Factorial of a given number 3 = 6	~

Write a C program to demonstrate functions without arguments and with return value. Correct The below code is used to check whether the given number is a prime number or not. Marked out of 1.00 Write the function prime(). Flag question Sample Input and Output: The given number is a prime number For example: Input Result The given number is a prime number 27 The given number is not a prime number The given number is not a prime number The given number is not a prime number

Answer: (penalty regime: 0 %)

Question 8

```
1 #include <stdio.h>
 2
 3 int prime(int num);
 5 int main()
 6 + {
        int num;
        scanf("%d",&num);
        if (prime(num) )
10 •
11
            printf("The given number is a prime number\n");
12
13
        else
14 .
            printf("The given number is not a prime number\n");
15
16
17
        return 0;
18
19
   // Write the function prime()
    int prime(int num)
22 - {
23
        if(num<=1)
24
        return 0:
        for(int i=2;i<=num/2;i++)</pre>
26 .
27
            if(num%i==0)
28 .
                return 0;
30
31
32
        return 1;
33
34
35
```

Reset answer

Input		Expected	Got	
	5	The given number is a prime number	The given number is a prime number	~
	27	The given number is not a prime number	The given number is not a prime number	~
	121	The given number is not a prime number	The given number is not a prime number	~
	1	The given number is not a prime number	The given number is not a prime number	~

Passed all tests! ✓

Finish review