S.No: 1	Exp. Name: ***Display hello world message***	Date: 2023-11-21

Aim:
Write a C program to display hello world message
Source Code:

hello.c

```
#include<stdio.h>
int main()
{
	char str[10];
	printf("Enter your name:");
	scanf("%s",&str);
	printf("Hello World\n");
	printf("Hello %s\n",str);
	return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Hello World
```

S.No: 2	Exp. Name: ***Scan all data type variables and display them***	**Date: 2023-11-21**

Aim:
Write a C program to scan all data type variables(int, float, char, double) as input and print them as output.

Input Format:
- First Line: An integer, entered after the prompt "**integer:** ".
- Second Line: A floating-point number, entered after the prompt "**floating-point number:** ".
- Third Line: A character, entered after the prompt "**character:** ".
- Fourth Line: A double-precision floating-point number, entered after the prompt "**double:** ".

Output Format:
- First Line: A message "**You entered:**".
- Second Line: The integer entered, in the format "**Integer: [integerVar]**".
- Third Line: The floating-point number entered, formatted to six decimal places, in the format "**Float: [floatVar]**".
- Fourth Line: The character entered, in the format "**Character: [charVar]**".
- Fifth Line: The double-precision floating-point number entered, formatted to six decimal places, in the format "**Double: [doubleVariable]**".

Note: Please add Space before %c which removes any white space (blanks, tabs, or newlines).
Source Code:

scan.c

```
#include<stdio.h>
#include<conio.h>
void main()
{
	int a;
	float b;
	char c;
	double d;
	printf("integer: ");
	scanf(" %d",&a);
	printf("floating-point number: ");
	scanf(" %f",&b);
	printf("character: ");
	scanf(" %c",&c);
	printf("double: ");
	scanf(" %lf",&d);
	printf("You entered:");
	printf("\nInteger: %d",a);
	printf("\nFloat: %f",b);
	printf("\nCharacter: %c",c);
	printf("\nDouble: %lf",d);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
integer:
9
floating-point number:
12.0254
character:
C
double:
12.02543124
You entered:
Integer: 9
Float: 12.025400
Character: C
Double: 12.025431
```

Test Case - 2

User Output

```
integer:
-10
floating-point number:
12.2546
character:
T
double:
12.6789678
You entered:
Integer: -10
Float: 12.254600
Character: T
Double: 12.678968
```

S.No: 3	Exp. Name: ***Arithmetic operations***	Date: 2023-11-21

Aim:
Write a C program to perform arithmetic operations like +,-,*,/,% on two input variables.

Input Format:
- The first line of input should be the value for first number
- The second line of input should be the value of second number

Output Format:
- The program prints the results of addition, subtraction, multiplication, division, and modulus

Note : For Division and Modulo operation, the value of num2 must be greater than 0
Source Code:

arithmeticOperations.c

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int n1,n2;
        printf("num1: ");
        scanf("%d",&n1);
        printf("num2: ");
        scanf("%d",&n2);
        printf("Sum: %d",(n1+n2));
        printf("\nDifference: %d",(n1-n2));
        printf("\nProduct: %d",(n1*n2));
        if(n2!=0)
        printf("\nDivision: %d",(n1/n2));
        else
        printf("\nInfinity");
        if(n2!=0)
        printf("\nModulus: %d\n",(n1%n2));
        else
        printf("\nModulo by zero is not allowed\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
num1:
9
num2:
8
Sum: 17
Difference: 1
Product: 72
Division: 1
```

```
Modulus: 1
```

Test Case - 2

User Output

```
num1:
1000
num2:
2
Sum: 1002
Difference: 998
Product: 2000
Division: 500
Modulus: 0
```

S.No: 4	Exp. Name: ***Write a C program to find Sum and Average of three numbers***	**Date: 2023-11-24**

Aim:

Write a program to find the `sum` and `average` of the three given integers.

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

Source Code:

Program314.c

```
#include<stdio.h>
void main()
{
	int a,b,c,sum;
	float avg;
	printf("Enter three integers : ");
	scanf("%d%d%d",&a,&b,&c);
	sum=a+b+c;
	avg=(float)sum/3;
	printf("Sum of %d, %d and %d : %d\n",a,b,c,sum);
	printf("Average of %d, %d and %d : %f\n",a,b,c,avg);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter three integers : 
121 34 56
Sum of 121, 34 and 56 : 211
Average of 121, 34 and 56 : 70.333336
```

Test Case - 2

User Output

```
Enter three integers : 
5 8 3
Sum of 5, 8 and 3 : 16
Average of 5, 8 and 3 : 5.333333
```

Test Case - 3

User Output

```
Enter three integers : 
-1 5 -6
Sum of -1, 5 and -6 : -2
Average of -1, 5 and -6 : -0.666667
```

S.No: 5	Exp. Name: ***Temperature conversions from Centigrade to Fahrenheit and vice versa.***	Date: 2023-11-28

Aim:
Write a C program to perform temperature conversions from Centigrade to Fahrenheit

Note : Refer to sample test cases for input and output format
Source Code:

temperature.c

```
#include<stdio.h>
void main()
{
        int a;
        float b,c,d,e;
        printf("Temperature Conversion:\n");
        printf("1.Celsius to Fahrenheit\n");
        printf("2.Fahrenheit to Celsius\n");
        printf("choice: ");
        scanf("%d",&a);
        switch(a)
                {
                        case 1 :
                        {
        printf("Enter Temperature in Celsius: ");
        scanf("%f",&b);
        c=32+b*9/5;
        printf("Fahrenheit Temperature: %.2f\n",c);
        break;
                        }
                        case 2:
                        {
                                printf("Enter Temperature in Fahrenheit: ");
                                scanf("%f",&d);
                                c=(d-32)*5/9;
                                printf("Celsius Temperature: %.2f\n",c);
                                break;
                        }
                        default:
                                {
                                        printf("Invalid choice\n");
                                }
                }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
37.5
37.50 Celsius = 99.50 Fahrenheit

Test Case - 2
User Output
-20
`-20.00 Celsius = -4.00 Fahrenheit`

S.No: 6	Exp. Name: ***Problem Solving***	**Date: 2023-11-21**

Aim:
Write a program to calculate the `simple interest` by reading **principle amount**, **rate of interest** and **time**.

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

```
Enter principle amount, rate of interest, time of loan :
```

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

```
Enter principle amount, rate of interest, time of loan : 23456.78 3.5 2.5
```

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

```
Simple Interest = 2052.468018
```

Note: Do use the **printf()** function and ensure that there is a `'\n'` at the end after print the result.

Source Code:

Program3.c

```
#include<stdio.h>
void main()
{
        float p,r,t,s;
        printf("Enter principle amount, rate of interest, time of loan : ");
        scanf("%f%f%f",&p,&r,&t);
        s=p* r* t/100;
printf("Simple Interest = %f\n",s);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter principle amount, rate of interest, time of loan :
2500 5 2
Simple Interest = 250.000000
```

S.No: 7	Exp. Name: ***Calculate the square root of an integer***	Date: 2023-11-24

Aim:
Write a program that prompts the user to enter an integer and calculates its square root.

Note: Print the result up to 3 decimal places.

Input format:
The program takes an integer as input with the print statement **"Enter an integer: "** followed by the integer.

Output format:
The output is the floating point value formatted to three decimals that represents the square root value of the user-given integer.

Hint: You can use **math** library to perform mathematical operations.

Instruction: During writing your code, please follow the input and output layout as mentioned in the sample test case.

Source Code:

squareRoot.c

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main()
{
int a;
float root;
printf("Enter an integer: ");
scanf("%d",&a);
root=sqrt(a);
printf("Square root: %.3f\n",root);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter an integer:
2
Square root: 1.414
```

Test Case - 2

User Output

```
Enter an integer:
4
Square root: 2.000
```

S.No: 8	Exp. Name: ***Calculate simple interest and compound interest***	**Date: 2023-11-28**

Aim:
Write a program to calculate the `simple interest` and `compound interest` by reading **principal amount**, **rate of interest** and **time**.

Note: Use the **printf()** function and ensure that the character `'\n'` is printed at the end of the result.

The formula to find simple interest is `simpleInterest = (principal * rate * time) / 100`.

The formula to find compound interest is
`compoundInterest = principal * pow(1 + (rate / 100), time) - principal`.

Note: Use `float` **data type** for all the involved variables.
Source Code:

Program315.c

```
#include<stdio.h>
#include<math.h>
int main()
{
	float P,R,T,SI,CI;
	printf("Enter P,R,T: ");
	scanf("%f%f%f",&P,&R,&T);
	SI=(P*R*T)/100;
	printf("SI= %F\n",SI);
	CI = P*pow(1+(R/100),T) - P;
	printf("CI= %f\n",CI);

}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter P,R,T:
5000 7 5
SI= 1750.000000
CI= 2012.760376
```

Test Case - 2

User Output

```
Enter P,R,T:
1000 6 4
SI= 240.000000
```

```
CI= 262.476685
```

S.No: 9	Exp. Name: ***Area of a triangle using heron's formula***	Date: 2023-11-28

Aim:

Write a program to find the `area` of a **triangle** using Heron's formula.

During execution, the program should print the following message on the console:

```
sides:
```

For example, if the user gives the following as **input** (input is positive floating decimal point numbers):

```
sides: 2.3 2.4 2.5
```

Then the program should **print** the result round off upto 2 decimal places as:

```
area: 2.49
```

Instruction: Your input and output layout must match with the sample test cases **(values as well as text strings)**.

The area of a triangle is given by `Area = √ p(p – a)(p – b)(p – c)`, where `p` is half of the perimeter, or `(a + b + c) / 2`. Let a,b,c be the lengths of the sides of the given triangle.

Hint: Use `sqrt` function defined in `math.h` header file

Source Code:

Program313.c

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
        float a,b,c,p,Area;
        printf("sides: ");
        scanf("%f%f%f",&a,&b,&c);
        p=(a+b+c)/2;
        Area=sqrt(p*(p-a)*(p-b)*(p-c));
        printf("area: %.2f\n", Area);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
sides:
2.3 2.4 2.5
area: 2.49
```

Test Case - 2

User Output

```
sides:
2.6 2.7 2.8
area: 3.15
```

S.No: 10	Exp. Name: ***Distance travelled by an object***	Date: 2023-11-28

Aim:
Write a program to find the `distance` travelled by an object.

Sample Input and Output:

```
Enter the acceleration value : 2.5
Enter the initial velocity : 5.7
Enter the time taken : 20
Distance travelled : 614.000000
```

Note - 1: Use the formula to find distance, `distance = ut + (1/2) at`^{2}.

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

Source Code:

DistanceTravelled.c

```
#include<stdio.h>
#include<conio.h>
void main()
{
	float u,a,s;
	int t;
	printf("Enter the acceleration value : ");
	scanf("%f",&a);
	printf("Enter the initial velocity : ");
	scanf("%f",&u);
	printf("Enter the time taken : ");
	scanf("%d",&t);
	s=(u*t)+(a*t*t)/2;
	printf("Distance travelled : %.6f\n",s);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter the acceleration value : 
4
Enter the initial velocity : 
5
Enter the time taken : 
6
Distance travelled : 102.000000
```

Test Case - 2

User Output

```
Enter the acceleration value :
5
Enter the initial velocity :
0
Enter the time taken :
10
Distance travelled : 250.000000
```

Test Case - 3

User Output

```
Enter the acceleration value :
2.5
Enter the initial velocity :
5.7
Enter the time taken :
20
Distance travelled : 614.000000
```

Test Case - 4

User Output

```
Enter the acceleration value :
50
Enter the initial velocity :
34.67
Enter the time taken :
6
Distance travelled : 1108.020020
```

Test Case - 5

User Output

```
Enter the acceleration value :
125.6
Enter the initial velocity :
45.8
Enter the time taken :
4
Distance travelled : 1188.000000
```

S.No: 11	Exp. Name: ***Evaluate the expressions***	Date: 2023-12-12

Aim:

Write a C program to evaluate the following expressions.
a. A+B*C+(D*E) + F*G
b. A/B*C-B+A*D/3
c. A+++B---A
d. J= (i++) + (++i)

Note: consider expression as A++ + ++B - --A

Source Code:

evaluate.c

```
#include<stdio.h>
int main()
{
	int A,B,C,D,E,F,G,i,a,b,c,d;
	printf("Enter values for A, B, C, D, E, F, G, i: ");
	scanf("%d%d%d%d%d%d%d%d",&A,&B,&C,&D,&E,&F,&G,&i);
	a=(A+B*C+(D*E)+F*G);
	printf("a.A+B*C+(D*E) + F*G = %d",a);
	printf("\n");
	b=(A/B*C-B+A*D/3);
	printf("b.A/B*C-B+A*D/3 = %d",b);
	printf("\n");
	c=(A++)+(B--)-A+2;
	printf("c.A+++B---A = %d",c);
	printf("\n");
	d=(i++)+(++i);
	printf("d.J = (i++) + (++i) = %d",d);
	printf("\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter values for A, B, C, D, E, F, G, i:
1 2 3 4 5 6 7 8
a.A+B*C+(D*E) + F*G = 69
b.A/B*C-B+A*D/3 = -1
c.A+++B---A = 3
d.J = (i++) + (++i) = 18
```

Test Case - 2

User Output

```
Enter values for A, B, C, D, E, F, G, i:
10 20 60 30 40 4 6 1
```

`a.A+B*C+(D*E) + F*G = 2434`
`b.A/B*C-B+A*D/3 = 80`
`c.A+++B---A = 21`
`d.J = (i++) + (++i) = 4`

S.No: 12	Exp. Name: ***Greatest of three numbers using a conditional operator***	**Date: 2023-12-12**

Aim:
Write a C program to display the greatest of three numbers using a conditional operator (ternary operator).

Input Format
The program prompts the user to enter three integers.

Output Format
The program prints the greatest of the three integers.

Source Code:

greatest.c

```
#include<stdio.h>
#include<conio.h>
int main()
{
        int a,b,c;
        printf("num1: ");
        scanf("%d",&a);
        printf("num2: ");
        scanf("%d",&b);
        printf("num3: ");
        scanf("%d",&c);
        if (a>=b&&a>c)
        {
                printf("Greatest number: %d\n",a);
        }
        else if (b>=a&&b>+c)
        {
                printf("Greatest number: %d\n",b);
        }
        else{
                printf("Greatest number: %d\n",c);
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
num1:
8
num2:
9
num3:
90
Greatest number: 90
```

Test Case - 2
User Output
`num1:`
5
`num2:`
45
`num3:`
6
`Greatest number: 45`

S.No: 13	Exp. Name: ***Write a C program to Total and Average of 5 subjects marks***	**Date: 2023-12-12**

Aim:

Write a program to take marks of `5` subjects in **integers**, and find the `total`, `average` in **float**.

Sample Input and Output:

```
Enter 5 subjects marks : 55 56 57 54 55
Total marks : 277.000000
Average marks : 55.400002
```

Note: Use the **printf()** function with a **newline** character (`\n`) to print the output at the end.

Source Code:

TotalAndAvg.c

```
#include<stdio.h>
int main()
{
        int a,b,c,d,e;
        float tm,am;
        printf("Enter 5 subjects marks : ");
        scanf("%d %d %d %d %d",&a,&b,&c,&d,&e);
        tm=(a+b+c+d+e);
        printf("Total marks : %f\n",tm);
        am=tm/5;
        printf("Average marks : %f",am);
        printf("\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter 5 subjects marks :
45 67 89 57 49
Total marks : 307.000000
Average marks : 61.400002
```

Test Case - 2

User Output

```
Enter 5 subjects marks :
55 56 57 54 55
Total marks : 277.000000
Average marks : 55.400002
```

Test Case - 3

User Output

```
Enter 5 subjects marks :
90 97 95 92 91
Total marks : 465.000000
Average marks : 93.000000
```

Test Case - 4

User Output

```
Enter 5 subjects marks :
20 30 66 77 44
Total marks : 237.000000
Average marks : 47.400002
```

Test Case - 5

User Output

```
Enter 5 subjects marks :
56 78 88 79 64
Total marks : 365.000000
Average marks : 73.000000
```

Test Case - 6

User Output

```
Enter 5 subjects marks :
44 35 67 49 51
Total marks : 246.000000
Average marks : 49.200001
```

S.No: 14	Exp. Name: ***Write a Program to find the Max and Min of Four numbers***	**Date: 2023-12-12**

Aim:

Write a program to find the `max` and `min` of **four** numbers.

Sample Input and Output :

```
Enter 4 numbers : 9 8 5 2
Max value : 9
Min value : 2
```

Note: Use the **printf()** function with a **newline** character (`\n`) to print the output at the end.

Source Code:

MinandMaxOf4.c

```
#include<stdio.h>
void main()
{
	int a,b,c,d;
	printf("Enter 4 numbers : ");
	scanf("%d%d%d%d",&a,&b,&c,&d);
	if(a>b&&a>c&&a>d)
	printf("Max value : %d\n",a);
	if(b>a&&b>c&&b>d)
		printf("Max value : %d\n",b);
	if(c>a&&c>b&&c>d)
		printf("Max value : %d\n",c);
	if(d>a&&d>b&&d>c)
		printf("Max value : %d\n",d);
	if(a<b&&a<c&&a<d)
		printf("Min value : %d\n",a);
	if(b<a&&b<c&&b<d)
		printf("Min value : %d\n",b);
	if(c<a&&c<b&&c<d)
		printf("Min value : %d\n",c);
	if(d<a&&d<b&&d<c)
		printf("Min value : %d\n",d);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter 4 numbers : 
9 8 5 2
Max value : 9
Min value : 2
```

Test Case - 2

User Output

```
Enter 4 numbers :
112 245 167 321
Max value : 321
Min value : 112
```

Test Case - 3

User Output

```
Enter 4 numbers :
110 103 113 109
Max value : 113
Min value : 103
```

Test Case - 4

User Output

```
Enter 4 numbers :
-34 -35 -24 -67
Max value : -24
Min value : -67
```

Test Case - 5

User Output

```
Enter 4 numbers :
24 28 34 16
Max value : 34
Min value : 16
```

Test Case - 6

User Output

```
Enter 4 numbers :
564 547 574 563
Max value : 574
Min value : 547
```

S.No: 15	Exp. Name: ***Find out the electricity bill charges***	Date: 2023-11-28

Aim:
An electricity board charges the following rates for the use of electricity:

- If units are less than or equal to 200, then the charge is calculated as 80 paise per unit.
- If units are less than or equal to 300, then the charge is calculated as 90 paise per unit.
- If units are beyond 300, then the charge is calculated as 1 Rupee per unit.

All users are charged a minimum of Rs. 100 as a meter charge even though the amount calculated is less than Rs. 100.

If the total amount charged is greater than Rs. 400, then an additional surcharge of 15% of the total amount is charged.

Write a C program to read the name of the user, and the number of units consumed and print out the charges as shown in the sample test cases.

Note: Print the amount charged up to 2 decimal places (actual amount, surcharges, amount to be paid).
Source Code:

```
electricityBillCharges.c
```

```
#include<stdio.h>
void main()
{
        char name[20];
        int u;
        float ac,sc,atp;
        printf("Enter customer name: ");
        scanf("%s",name);
        printf("Units consumed: ");
        scanf("%d",&u);
        if (u<=200)
        {
                ac=u*0.80;
        }
        else if (u<=300)
        {
                ac=u*0.90;
        }
                        else
                        {
                                ac=u*1;
                        }
                        {
                                if(ac>400)
                                {
                                        sc=0.15*ac;
                                        atp=sc+ac;
                                }
                                else if (ac<100)
                                {
                                        atp=100;
                                }
                                else
                                {
                                        atp=ac;
                                }
                        }
                        printf("Customer name: %s\n",name);
                        printf("Units consumed: %d\n",u);
                        printf("Amount charged: %.2f\n",ac);
                        printf("Surcharges: %.2f\n",sc);
                        printf("Amount to be paid: %.2f\n",atp);
                }
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output

```
Enter customer name:
John
Units consumed:
78
```

```
Customer name: John
Units consumed: 78
Amount charged: 62.40
Surcharges: 0.00
Amount to be paid: 100.00
```

Test Case - 2

User Output

```
Enter customer name:
Rosy
Units consumed:
325
Customer name: Rosy
Units consumed: 325
Amount charged: 325.00
Surcharges: 0.00
Amount to be paid: 325.00
```

Test Case - 3

User Output

```
Enter customer name:
Amar
Units consumed:
801
Customer name: Amar
Units consumed: 801
Amount charged: 801.00
Surcharges: 120.15
Amount to be paid: 921.15
```

Test Case - 4

User Output

```
Enter customer name:
Raman
Units consumed:
300
Customer name: Raman
Units consumed: 300
Amount charged: 270.00
Surcharges: 0.00
Amount to be paid: 270.00
```

S.No: 16	Exp. Name: ***C program to find roots and nature of quadratic equation.***	**Date: 2023-11-28**

Aim:
Write a C program to find the roots of a quadratic equation, given its coefficients.
Source Code:

quad.c

```
#include<stdio.h>
#include<math.h>
void main()
{
double a,b,c,d,root1,root2,realpart,imagpart;
        printf("Enter coefficients a, b and c: ");
        scanf("%lf%lf%lf",&a,&b,&c);
        d=(b*b)-4*a*c;
        if(d>0)
        {
                root1=(-b+sqrt(d))/(2*a);
                root2=(-b-sqrt(d))/(2*a);
        }
        else if(d==0)
        {
                root1=-b/2*a;
                root2=-b/2*a;
        }
        else
        {
                realpart=-b/(2*a);
                imagpart=sqrt(-d)/(2*a);
        }
        printf("root1 = %.2lf+%.2lfi and root2 = %.2lf-
%.2lfi",realpart,imagpart,realpart,imagpart);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter coefficients a, b and c:
3 7 9
root1 = -1.17+1.28i and root2 = -1.17-1.28i
```

Test Case - 2

User Output

```
Enter coefficients a, b and c:
8 8 6
root1 = -0.50+0.71i and root2 = -0.50-0.71i
```

S.No: 17	Exp. Name: ***Simulate a basic calculator***	Date: 2023-11-28

Aim:
Write a program to perform basic calculator operations **[+, -, *, /]** of two integers **a** and **b** using switch statement.

Constraints:
- 10^{-4} <= a,b = 10^{4}
- operations allowed are +, -, *, /
- "/" divisibility will perform integer division operation.

Input Format: The first line of the input consists of an integer which corresponds to **a**, character which corresponds to the **operator** and an integer which corresponds to **b**.

Output format: Output consists of result after performing mentioned operation (a operation b).

Instruction: To run your custom test cases strictly map your input and output layout with the visible test cases.
Source Code:

calculator.c

```
#include<stdio.h>
void main()
{
        int a,b,add,sub,multi,div;
        char oper;

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

        switch(oper)
                {
                        case '+':
                        add=a+b;
                        printf("%d\n",add);
                        break;
                                case '-':
                        sub=a-b;
                        printf("%d\n",sub);
                        break;
                        case '*':
                                multi=a*b;
                        printf("%d\n",multi);
                        break;
                        case '/':
                                div=a/b;
                        printf("%d\n",div);
                        break;
                }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
36-31
5

Test Case - 2
User Output
89/45
1

Test Case - 3
User Output
10000/10000
1

S.No: 18	Exp. Name: ***Write a program to find leap year or not***	**Date: 2023-12-14**

Aim:
Lucy is celebrating her 15th birthday. Her father promised her that he will buy her a new computer on her birthday if she solves the question asked by him.

He asks Lucy to find whether the year on which she had born is **leap year or not**.

Help her to solve this puzzle so that she celebrates her birthday happily. If her birth year is 2016 and it is a leap year display 2016 is a leap year.? Else display 2016 is not a leap year and check with other leap year conditions.

Source Code:

leapYear.c

```
#include<stdio.h>
void main()
{
	int year;
	scanf("%d",&year);
	if(year%4==0 && year %100 !=0)
		printf("%d is a leap year\n",year);
	else
		printf("%d is not a leap year\n",year);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
1900
1900 is not a leap year
```

Test Case - 2

User Output

```
2004
2004 is a leap year
```

Test Case - 3

User Output

```
1995
1995 is not a leap year
```

S.No: 19	Exp. Name: ***Factorial of a given number***	Date: 2023-12-12

Aim:

Write a C program to find the factorial of a given number

Source Code:

factorialOfInt.c

```
#include<stdio.h>
void main()
{
        int n, i;
        unsigned long long fact =1;
        printf("Integer: ");
        scanf("%d",&n);
        if(n<0)
                printf("Error! Factorial of a negative number doesn't exist:");
        else
        {
                for(i= 1; i<=n;i++)
                        {
                                fact *=i;
                        }
                printf("Factorial: %llu\n", fact);
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Integer: 
5
Factorial: 120
```

Test Case - 2

User Output

```
Integer: 
4
Factorial: 24
```

S.No: 20	Exp. Name: ***C program to determine whether a given number is prime or not.***	Date: 2023-12-12

Aim:
Write the C program to determine whether a given number is prime or not.
Source Code:

Prime.c

```
#include<stdio.h>
int isprime(int num)
{
        if (num <= 1)
        {
                return 0;
        }
        for (int i = 2; i <= num /2; i++)
                {
                        if (num % i == 0)
                        {
                                return 0;
                        }
                }
        return 1;
}
int main()
{
        int number;
        printf("Enter a number: ");
        scanf("%d", &number);
        if (isprime(number))
        {
                printf("%d is a prime number\n", number);
}
        else
        {
                printf("%d is not a prime number\n", number);
        }
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter a number:
9
9 is not a prime number
```

Test Case - 2

User Output

```
Enter a number:
11
11 is a prime number
```

S.No: 21	Exp. Name: ***Compute sine and cos series using taylor series***	Date: 2023-12-14

Aim:
Write a C program to compute the sine and cosine series using the Taylor series.

Taylor series:

sin x = x - (x^3/3!) + (x^5/5!) - (x^7/7!) + ...

cos x = 1 - (x^2/2!) + (x^4/4!) - (x^6/6!) + ...

Note: Print the result up to 4 decimal places. Use the **double** data type for all variables except for the number of terms in the series, which should be an integer. Additionally, initialize the variables that will store the results of the sine and cosine series to **0.0** at the beginning.

Source Code:

taylor.c

```
#include <stdio.h>
#include <math.h>
void main()
{
        int terms,fact=1;
        float x,term1=1,term2=1,sine,cosine;
        printf("angle in radians: ");
        scanf("%f",&x);
        printf("number of terms in the series: ");
        scanf("%d",&terms);
        sine=x;
        cosine=1;
        for(int i=1;i<terms;i++)
                {
                        term1 = pow(-1,i)*pow(x,2*i);
                                fact *= (2*i);
                        cosine += term1/fact;
                        term2 = pow(-1,i)*pow(x,2*i+1);
                                fact *= (2*i+1);
                        sine += term2/fact;
                }
        printf("Sine = %.4f\n",sine);
        printf("Cosine = %.4f\n",cosine);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
angle in radians:
0.5
number of terms in the series:
```

```
3
Sine = 0.4794
Cosine = 0.8776
```

Test Case - 2

User Output

```
angle in radians:
0.6
number of terms in the series:
5
Sine = 0.5646
Cosine = 0.8253
```

S.No: 22	Exp. Name: ***Check given number is palindrome or not***	**Date: 2023-12-12**

Aim:
Write an C program to check given number is palindrome or not

Input Format:
- Single Line: An integer value representing the number to be checked for palindrome status.

Output Format:
- Single Line: A message indicating whether the number is a palindrome or not. The format of the message will be:
- "**[number] is a palindrome.**" if the number is a palindrome.
- "**[number] is not a palindrome.**" if the number is not a palindrome.

Source Code:

palindrome.c

```
#include<stdio.h>
int main()
{
        int n, reversed =0, remainder, original;
        printf("");
        scanf("%d", &n);
        original =n;
        while (n != 0)
                {
                        remainder = n % 10;
                        reversed = reversed * 10 + remainder;
                        n /= 10;
                }
        if(original == reversed )
        {
                printf("%d is a palindrome.\n",original);
        }
        else
        {
                printf("%d is not a palindrome.\n",original);
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
121
121 is a palindrome.
```

Test Case - 2

User Output

```
143
```

```
143 is not a palindrome.
```

S.No: 23	Exp. Name: ***Pyramid with numbers***	Date: 2023-12-12

Aim:
Write a program to print a `pyramid` of **numbers** separated by spaces for the given number of rows.

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

```
Enter number of rows : 
```

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

```
Enter number of rows : 3
```

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

```
  1
 1 2
1 2 3
```

Source Code:

PyramidDemo15.c

```
#include <stdio.h>
void main() {
        int n, i, j, s;
        printf("Enter number of rows : ");
        scanf("%d", &n);
        // Fill the missing code

        for (i = 1; i <= n; ++i)
                {
                        for (s = 1; s <= n - i; ++s)
                                printf(" ");
                        for (j = 1; j <= i;++j)
                                printf("%d ", j);
                        for (j = i - 1; j >= 1; --j)
                        for (j = i - 1; j >= 1; --j);
                        printf("\n");
}

}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter number of rows : 
3
  1
 1 2
```

```
1 2 3
```

Test Case - 2

User Output

```
Enter number of rows :
6
     1
    1 2
   1 2 3
  1 2 3 4
 1 2 3 4 5
1 2 3 4 5 6
```

Test Case - 3

User Output

```
Enter number of rows :
8
       1
      1 2
     1 2 3
    1 2 3 4
   1 2 3 4 5
  1 2 3 4 5 6
 1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
```

S.No: 24	Exp. Name: ***Minimum and maximum in an array of integers.***	**Date: 2023-12-19**

Aim:
Write a C program to find the **minimum** and **maximum** in an array of integers.
Source Code:

ArrayElements.c

```
#include <stdio.h>
void main() {
	int arr[20], number, 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 and minimum in the given integer 
array*/
	min=arr[0];
	max=arr[0];
	for(int i=1;i<number;i++)
		{
			if(arr[i]>max) max=arr[i];
			if(arr[i]<min) min=arr[i];
		}
	printf("Min an Max: %d and %d",min,max );
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
5
Elements:
4 9 6 8 2
Min an Max: 2 and 9
```

Test Case - 2

User Output

```
1
Elements:
216
Min an Max: 216 and 216
```

S.No: 25	Exp. Name: ***Search for an element using Linear search***	**Date: 2023-12-19**

Aim:
Write a C program to check whether the given element is present or not in the array of elements using linear search.

Source Code:

SearchEle.c

```
#include<stdio.h>
void main()
{
	int arr[100], number , snumber,i;
	printf("Enter size: ");
	scanf("%d", & number);
	printf("Enter %d element: ",number);
	for(int i=0;i<number;i++){
		scanf("%d", &arr[i]);
	}
	printf("Enter search element: ");
	scanf("%d",&snumber);
	for(i=0;i<number;i++)
		if(arr[i]==snumber)
			break;
	if(i==number)
		printf("%d is not found\n",snumber);
	else
		printf("Found at position %d\n",i);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter size:
6
Enter 6 element:
2 4 8 1 3 5
Enter search element:
6
6 is not found
```

Test Case - 2

User Output

```
Enter size:
6
Enter 6 element:
```

```
2 4 8 1 3 5
Enter search element:
2
Found at position 0
```

Test Case - 3

User Output

```
Enter size:
6
Enter 6 element:
2 4 8 1 3 5
Enter search element:
9
9 is not found
```

S.No: 26	Exp. Name: ***C program to reverse an array.***	**Date: 2023-12-12**

Aim:

Write a C program to reverse the elements an array of integers.

Source Code:

reverseArray.c

```
#include<stdio.h>
int main()
{
	int arr[50];
	int i,size;
	printf("Enter no of elements: ");
	scanf("%d",&size);
	printf("Enter elements: ");
	for(i=0;i<size;i++){
		scanf("%d",&arr[i]);
	}
	printf("The reversed array: ");
	for(i=size-1;i>=0;i--)
	{
		printf("%d ",arr[i]);;
	}
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter no of elements:
5
Enter elements:
3 4 1 2 4
The reversed array: 4 2 1 4 3
```

Test Case - 2

User Output

```
Enter no of elements:
8
Enter elements:
2 5 1 77 33 88 2 9
The reversed array: 9 2 88 33 77 1 5 2
```

S.No: 27	Exp. Name: ***2's complement of a given Binary number***	**Date: 2023-12-19**

Aim:
Write a **C** program to find 2's complement of a given binary number.

Note: The binary input should be separated by a **space**.

Source Code:

twosComplement.c

```
#include<stdio.h>
int main()
{
	int arr[50],number,flag=0;
	printf("Enter size: ");
	scanf("%d",&number);
	printf("Enter %d bit binary number: ",number);
	for(int i=0;i<number;i++)
		scanf("%d",&arr[i]);
	for(int i=number;i>=0;i--)
		if(flag==0)
		{if(arr[i]==1) flag=1;}
	else
		{if(arr[i]==1) arr[i]=0; else arr[i]=1;}
	printf("2\'s complement: ");
	for(int i=0;i<number;i++)
		printf("%d ", arr[i]);
	printf("\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter size:
5
Enter 5 bit binary number:
1 0 0 1 0
2's complement: 0 1 1 1 0
```

Test Case - 2

User Output

```
Enter size:
6
Enter 6 bit binary number:
1 0 0 0 1 1
2's complement: 0 1 1 1 0 1
```

S.No: 28	Exp. Name: ***Eliminate duplicate elements in an array***	**Date: 2023-12-19**

Aim:
Write a C program to eliminate duplicate elements of an array.

Input Format:
- First Line: An integer n representing the size of the array.
- Second Line: n integers representing the elements of the array.

Output Format:
- Single Line: A space-separated list of the unique elements of the array after duplicates have been removed.

Source Code:

eliminateDuplicates.c

```
#include<stdio.h>
int main()
{
        int arr[50], number, match;
        printf("Enter size: ");
        scanf("%d", &number);
        printf("Enter %d elements: ", number);
        for(int i=0;i<number;i++)
                scanf("%d", &arr[i]);
        printf("After eliminating duplicates: ");
        for(int i=0;i<number;i++)
                if(i==0) printf("%d ",arr[i]);
        else
                {match=0;
                        for(int j=0;j<i;j++)
                                if(arr[i]==arr[j]) {match=1;break;}
                 if(match==0)
                         printf("%d ",arr[i]);
                        }
        printf("\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter size:
5
Enter 5 elements:
1 2 1 2 3
After eliminating duplicates: 1 2 3
```

Test Case - 2

User Output

```
Enter size:
5
Enter 5 elements:
11 13 11 12 13
After eliminating duplicates: 11 13 12
```

S.No: 29	Exp. Name: ***Addition of Two Matrices***	Date: 2023-12-19

Aim:
Write a C program to perform the addition of two matrices.

Input Format:
The first line contains two space separated integers, row & col, representing the number of rows and columns of each matrix
The second line contains row * col number of space separated integers representing the elements of matrix 1
The last line contains row * col number of space separated integers representing the elements of matrix 2

Output Format:
row number of lines with col number of space separated elements representing the elements of sum matrix

Note: Addition of two matrices can only be done when the dimensions of both matrices are same, so we are taking the same dimensions for both matrices.

Source Code:

addTwoMatrices.c

```
#include<stdio.h>
int main()
{
	int r,c,matrix1[10][10],matrix2[10][10];
	printf("Enter no of rows, columns: ");
	scanf("%d%d",&r,&c);
	printf("Elements of matrix 1: ");
	for(int i=0;i<r;i++)
		for(int j=0;j<c;j++)
			scanf("%d",&matrix1[i][j]);
			printf("Elements of matrix 2: ");
				for(int i=0;i<r;i++)
		for(int j=0;j<c;j++)
	scanf("%d",&matrix2[i][j]);
	printf("Addition of matrices:\n");
	for(int i=0;i<r;i++)
		{
		for(int j=0;j<c;j++)
			printf("%d ",matrix1[i][j]+matrix2[i][j]);
		printf("\n");
		}
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter no of rows, columns:
1 2
Elements of matrix 1:
1 2
```

```
Elements of matrix 2:
9 8
Addition of matrices:
10 10
```

Test Case - 2

User Output

```
Enter no of rows, columns:
2 3
Elements of matrix 1:
1 2 3 4 5 6
Elements of matrix 2:
9 8 7 6 5 4
Addition of matrices:
10 10 10
10 10 10
```

S.No: 30	Exp. Name: ***Multiplication of two matrices***	**Date: 2023-12-19**

Aim:
Write a C program to find the multiplication of two matrices

Input Format:

- First line contains an integer r and an integer c, representing the number of rows and columns
- Next r rows contains c number of integers representing the elements of the matrix1
- Repeat the Same for matrix2

Output Format:

- Prints the matrix1 and matrix2 and finally the result of multiplication of both the matrices

Note: For more clarification refer to the shown test cases

Source Code:

`matrixMul.c`

```
#include<stdio.h>
int main()
{
	int r1,c1,r2,c2,matrix1[10][10],matrix2[10][10],matrix3[10][10],sum;
	printf("no of rows, columns of matrix1: ");
	scanf("%d%d",&r1,&c1);
	printf("matrix1 elements:\n");
	for(int i=0;i<r1;i++)
		for(int j=0;j<c1;j++)
			scanf("%d",&matrix1[i][j]);
	printf("no of rows, columns of matrix2: ");
	scanf("%d%d",&r2,&c2);
	printf("matrix2 elements:\n");
	for(int i=0;i<r2;i++)
		for(int j=0;j<c2;j++)
			scanf("%d",&matrix2[i][j]);
	printf("Given matrix1:\n");
	for(int i=0;i<r1;i++)
		{
			for(int j=0;j<c1;j++)
				printf("%d ",matrix1[i][j]);
			printf("\n");
		}
	printf("Given matrix2:\n");
	for(int i=0;i<r2;i++)
		{
			for(int j=0;j<c2;j++)
				printf("%d ",matrix2[i][j]);
			printf("\n");
		}
	if(c1!=r2)
		printf("Multiplication not possible\n");
	else{
		printf("Multiplication of two matrices:\n");
		for(int i=0;i<r1;i++)
			{
				for(int j=0;j<c2;j++)
					{
						matrix3[i][j]=0;
						for(int k=0;k<c1;k++)
		matrix3[i][j]+= matrix1[i][k] * matrix2[k][j];
						printf("%d ", matrix3[i][j]);
					}
				printf("\n");
			}
	}
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
no of rows, columns of matrix1:
```

```
2 2
matrix1 elements:
11 22
33 44
no of rows, columns of matrix2:
2 2
matrix2 elements:
11 22
33 44
Given matrix1:
11  22
33  44
Given matrix2:
11  22
33  44
Multiplication of two matrices:
847  1210
1815  2662
```

Test Case - 2

User Output

```
no of rows, columns of matrix1:
3 3
matrix1 elements:
1 2 3
4 5 6
7 8 9
no of rows, columns of matrix2:
2 3
matrix2 elements:
1 2 3
4 5 6
Given matrix1:
1  2  3
4  5  6
7  8  9
Given matrix2:
1  2  3
4  5  6
Multiplication not possible
```

S.No: 31	Exp. Name: ***Write a C program to Sort given elements using Bubble sort***	Date: 2023-12-19

Aim:

Develop an algorithm, implement and execute a **C** program that reads **n** integer numbers and arrange them in **ascending order** using **Bubble Sort**.

Source Code:

Lab7.c

```
#include<stdio.h>
int main()
{
	int i,j,n,arr[20],temp;
	scanf("%d",&n);
	printf("Elements: ");
	for(i=0;i<n;i++)
		scanf("%d",&arr[i]);
	printf("Before sorting: ");
	for(i=0;i<n;i++)
		printf("%d ",arr[i]);
	printf("\n");
	for(i=0;i<n-1;i++)
		for(j=0;j<n-i-1;j++)
			if(arr[j]>arr[j+1])
			{
				int temp = arr[j];
				arr[j] = arr[j+1];
				arr[j+1]= temp;
			}
	printf("After sorting: ");
	for(int i=0;i<n;i++)
		printf("%d ",arr[i]);
	printf("\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
4
Elements:
44 22 66 11
Before sorting: 44 22 66 11
After sorting: 11 22 44 66
```

Test Case - 2

User Output

```
5
```

```
Elements:
9 2 7 1 6
Before sorting: 9 2 7 1 6
After sorting: 1 2 6 7 9
```

S.No: 32	Exp. Name: ***Concatenate two given strings without using string library functions***	**Date: 2023-12-19**

Aim:
Write a program to `concatenate` two given strings without using string library functions.

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

```
string1 :
```

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

```
string1 : ILove
```

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

```
string2 :
```

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

```
string2 : Coding
```

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

```
concatenated string = ILoveCoding
```

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

Source Code:

Program605.c

```
#include<stdio.h>
void main()
{
	char str1[20],str2[20];
	printf("string1 : ");
	scanf("%s",str1);
	printf("string2 : ");
	scanf("%s",str2);
	printf("concatenated string = %s%s\n",str1,str2);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
string1 :
ILove
string2 :
Coding
concatenated string = ILoveCoding
```

Test Case - 2

User Output

```
string1 :
1234
string2 :
567
concatenated string = 1234567
```

S.No: 33	Exp. Name: ***Reverse the given string without using the library functions***	Date: 2023-12-19

Aim:

Write a program to `reverse` the given string without using the library functions.

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

```
Enter a string :
```

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

```
Enter a string : Dallas
```

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

```
Reverse string : sallaD
```

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

Source Code:

Program609.c

```
#include<stdio.h>
int main()
{
        char str1[20],len;
        int i;
        printf("Enter a string : ");
        scanf("%s",str1);
        len=0;
        while(str1[len]!='\0')
                {
                        len++;
                }
        printf("Reverse string : ");
        for(i=len-1;i>=0;i--)
                printf("%c",str1[i]);
        printf("\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter a string :
Dallas
Reverse string : sallaD
```

S.No: 34	Exp. Name: ***Write a C program to find Sum of array elements by allocating memory using malloc() function***	Date: 2023-12-28

Aim:
Write a program to find the `sum` of n elements by allocating memory by using **malloc()** function.

Note: Write the functions **allocateMemory()**, **read1()** and **sum()** in `UsingMalloc.c`

Source Code:

SumOfArray1.c

```
#include <stdio.h>
#include <stdlib.h>
#include "UsingMalloc.c"
void main() {
	int *p, n, i;
	printf("Enter n value : ");
	scanf("%d", &n);
	p = allocateMemory(n);
	printf("Enter %d values : ", n);
	read1(p, n);
	printf("The sum of given array elements : %d\n", sum(p, n));
}
```

UsingMalloc.c

```
int *allocateMemory(int n)
{
	int *p;
	p=malloc(n * sizeof(int));
	return p;
}
void read1(int *p,int n)
{
	for(int i=0;i<n;i++)
		{
			scanf("%d",(p+i));
		}
}
int sum(int *p,int n)
{
	int total=0;
	for(int i=0;i<n;i++)
		{
			total+=*(p+i);
		}
	return total;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter n value :
3
Enter 3 values :
10 20 30
The sum of given array elements : 60
```

Test Case - 2

User Output

```
Enter n value :
4
Enter 4 values :
-5 -6 -4 -2
The sum of given array elements : -17
```

S.No: 35	Exp. Name: ***Write a program to find Total and Average gained by Students in a Section using Array of Structures***	Date: 2024-01-06

Aim:

Write a **C** program to find out the `total` and `average marks` gained by the students in a section using **array of structures**.

Note: Consider that regdno, marks of 3 subjects, total and average are the members of a structure and make sure to provide the int value for **number of students** which are lessthan `60`

Sample Input and Output:

```
Enter number of students : 3
Enter regdno, three subjects marks of student-0: 101 56 78 76
Enter regdno, three subjects marks of student-1: 201 76 89 91
Enter regdno, three subjects marks of student-2: 301 46 57 61
Student-0 Regdno = 101  Total marks = 210   Average marks = 70.000000
Student-1 Regdno = 201  Total marks = 256   Average marks = 85.333336
Student-2 Regdno = 301  Total marks = 164   Average marks = 54.666668
```

Source Code:

ArrayOfStructures2.c

```
#include <stdio.h>

struct student {

// Write the members of structure

int regdno;

int marks[3];

};

void main() {

struct student s[60];

int i, n, total;

float average;

printf("Enter number of students : ");

scanf("%d", &n);

for (i=0;i<n;i++) { // Complete the code in for

printf("Enter regdno, three subjects marks of student-%d: ", i); // Read regdno and 3
subjects marks

scanf("%d%d%d%d",&s[i].regdno, &s[i].marks[0],&s[i].marks[1],&s[i].marks[2]);

}

for (i=0;i<n;i++) { // Complete the code in for

// Find Total and Average

total=s[i].marks[0]+s[i].marks[1]+s[i].marks[2];

average=total/3.0;

printf("Student-%d Regdno = %d\tTotal marks = %d\tAverage marks =
%f\n",i,s[i].regdno,total,average);
}
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
`Enter number of students :`
3

```
Enter regdno, three subjects marks of student-0:
101 56 78 76
Enter regdno, three subjects marks of student-1:
201 76 89 91
Enter regdno, three subjects marks of student-2:
301 46 57 61
Student-0 Regdno = 101  Total marks = 210       Average marks = 70.000000
Student-1 Regdno = 201  Total marks = 256       Average marks = 85.333336
Student-2 Regdno = 301  Total marks = 164       Average marks = 54.666668
```

Test Case - 2

User Output

```
Enter number of students :
10
Enter regdno, three subjects marks of student-0:
501 23 45 67
Enter regdno, three subjects marks of student-1:
502 78 65 76
Enter regdno, three subjects marks of student-2:
503 99 87 67
Enter regdno, three subjects marks of student-3:
504 89 78 82
Enter regdno, three subjects marks of student-4:
505 37 59 76
Enter regdno, three subjects marks of student-5:
506 78 59 67
Enter regdno, three subjects marks of student-6:
507 92 72 82
Enter regdno, three subjects marks of student-7:
508 45 47 48
Enter regdno, three subjects marks of student-8:
509 55 52 59
Enter regdno, three subjects marks of student-9:
510 62 61 66
Student-0 Regdno = 501  Total marks = 135       Average marks = 45.000000
Student-1 Regdno = 502  Total marks = 219       Average marks = 73.000000
Student-2 Regdno = 503  Total marks = 253       Average marks = 84.333336
Student-3 Regdno = 504  Total marks = 249       Average marks = 83.000000
Student-4 Regdno = 505  Total marks = 172       Average marks = 57.333332
Student-5 Regdno = 506  Total marks = 204       Average marks = 68.000000
Student-6 Regdno = 507  Total marks = 246       Average marks = 82.000000
Student-7 Regdno = 508  Total marks = 140       Average marks = 46.666668
Student-8 Regdno = 509  Total marks = 166       Average marks = 55.333332
Student-9 Regdno = 510  Total marks = 189       Average marks = 63.000000
```

Test Case - 3

User Output

```
Enter number of students :
5
Enter regdno, three subjects marks of student-0:
101 76 78 73
Enter regdno, three subjects marks of student-1:
102 89 57 68
Enter regdno, three subjects marks of student-2:
103 77 67 59
Enter regdno, three subjects marks of student-3:
104 37 47 52
Enter regdno, three subjects marks of student-4:
105 88 47 69
Student-0 Regdno = 101  Total marks = 227       Average marks = 75.666664
Student-1 Regdno = 102  Total marks = 214       Average marks = 71.333336
Student-2 Regdno = 103  Total marks = 203       Average marks = 67.666664
Student-3 Regdno = 104  Total marks = 136       Average marks = 45.333332
Student-4 Regdno = 105  Total marks = 204       Average marks = 68.000000
```

S.No: 36	Exp. Name: ***Write a Program to enter n students data using calloc() and display Failed Students List***	**Date: 2024-01-06**

Aim:
Write a **C** program to enter **n** students' data using **calloc()** and display the **students list**.

Note: If marks are less than 35 in any subject, the student will fail
Source Code:

FailedList.c

```
#include <stdio.h>
#include <stdlib.h>
struct student {
	int roll;
	int marks[6], sum;
	float avg;
};
#include "FailedList1.c"
void main() {
	struct student *s;
	int i, n;
	printf("Enter the number of students : ");
	scanf("%d", &n);
	s = allocateMemory(s, n);
	read1(s, n);
	calculateMarks(s, n);
	displayFailedList(s, n);
}
```

FailedList1.c

```
struct student* allocateMemory(struct student *s, int n) {
	// Write the code
	struct student *p;
	p=(struct student *)calloc(n,sizeof(struct student));
	return p;
}

void read1(struct student *s, int n) {
	// write the code
	for(int i=0;i<n;i++)
		{
			printf("Enter the details of student - %d\n",i+1);
			printf("Enter the roll number : ");
			scanf("%d",&(s+i)->roll);
			printf("Enter 6 subjects marks : ");
			for(int j=0;j<6;j++)
				{
					scanf("%d",&(s+i)->marks[j]);
				}
		}
}
void calculateMarks(struct student *s, int n) {
	// write the code
	for(int i=0;i<n;i++)
		{
			(s+i)->sum = 0;
			for(int j=0;j<6;j++)
				{
					(s+i)->sum = (s+i)->sum + (s+i)->marks[j] ;
				}
			(s+i)->avg = (s+i)->sum /6.0;
		}
}
void displayFailedList(struct student *s, int n) {
	int i;
	printf("RollNo\tTotalMarks\tAverageMarks\tStatus\n");
	for (i = 0; i < n; i++) {
		printf("%d\t", (s+i)->roll); // Fill the missing code
		printf("%d\t", (s+i)->sum); // Fill the missing code
		printf("%f\t", (s+i)->avg); // Fill the missing code
		if ((s+i)->marks[0]<35 || (s+i)->marks[1]<35 ||(s+i)->marks[2]<35 ||(s+i)-
>marks[3]<35 ||(s+i)->marks[4]<35 ||(s+i)->marks[5]<35 )
			printf("Fail");
		else
			printf("Pass");
		printf("\n");
	}
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output

```
Enter the number of students :
3
Enter the details of student - 1
Enter the roll number :
101
Enter 6 subjects marks :
45 67 58 36 59 63
Enter the details of student - 2
Enter the roll number :
102
Enter 6 subjects marks :
34 56 98 39 78 89
Enter the details of student - 3
Enter the roll number :
103
Enter 6 subjects marks :
35 67 89 98 76 56
RollNo  TotalMarks      AverageMarks    Status
101     328     54.666668       Pass
102     394     65.666664       Fail
103     421     70.166664       Pass
```

Test Case - 2

User Output

```
Enter the number of students :
2
Enter the details of student - 1
Enter the roll number :
1001
Enter 6 subjects marks :
26 57 68 67 67 65
Enter the details of student - 2
Enter the roll number :
1002
Enter 6 subjects marks :
58 67 58 89 87 76
RollNo  TotalMarks      AverageMarks    Status
1001    350     58.333332       Fail
1002    435     72.500000       Pass
```

S.No: 37	Exp. Name: ***Write a C program to find Total Marks of a Student using Command-line arguments***	**Date: 2024-01-06**

Aim:

Write a C program to read student name and **3** subjects marks from the **command line** and display the student details along with total.

Sample Input and Output - 1:

```
If the arguments passed as $./TotalMarksArgs.c Sachin 67 89 58, then the program should
print the output as:
Cmd Args : Sachin 67 89 58
Student name : Sachin
Subject-1 marks : 67
Subject-1 marks : 89
Subject-1 marks : 58
Total marks : 214
```

Sample Input and Output - 2:

```
If the arguments passed as $./TotalMarksArgs.c Johny 45 86 57 48, then the program should
print the output as:
Cmd Args : Johny 45 86 57 48
Arguments passed through command line are not equal to 4
```

Hint : `atoi()` is a library function that converts string to integer. When program gets the input from command line, string values transfer in the program, we have to convert them to integers. `atoi()` is used to return the integer of the string arguments.

Source Code:

TotalMarksArgs.c

```c
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char* argv[])
{
	if(argc != 5)
		printf("Arguments passed through command line are not equal to 4\n");
	else
	{
		printf("Student name : %s\n",argv[1]);
		printf("Subject-1 marks : %s\n",argv[2]);
		printf("Subject-2 marks : %s\n",argv[3]);
		printf("Subject-3 marks : %s\n",argv[4]);
		printf("Total marks : %d\n",atoi(argv[2])+atoi(argv[3])+atoi(argv[4]));
	}
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Student name : Sachin
Subject-1 marks : 67
Subject-2 marks : 89
Subject-3 marks : 58
Total marks : 214
```

Test Case - 2

User Output

```
Arguments passed through command line are not equal to 4
```

S.No: 38	Exp. Name: ***Write a C program to implement realloc()***	**Date: 2024-01-06**

Aim:

Write a **C** program to implement `realloc()`.

The process is

1. Allocate memory of an array with size 2 by using malloc()
2. Assign the values 10 and 20 to the array
3. Reallocate the size of the array to 3 by using realloc()
4. Assign the value 30 to the newly allocated block
5. Display all the 3 values

Source Code:

ProgramOnRealloc.c

```
#include <stdio.h>
#include <stdlib.h>
int main() {
	int *ptr = (int *)malloc(sizeof(int) * 2);
	int i;
	int *ptr_new;
	*ptr = 10;
	*(ptr + 1) = 20;
	// Reallocate the *ptr size to 3
	ptr_new=(int*) realloc(ptr, 3 * sizeof(int));
	//Assign the value 30 to newly allocated memory
	*(ptr_new + 2) = 30;
	for (i = 0; i < 3; i++)
		printf("%d ", *(ptr_new + i));
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
10 20 30
```

S.No: 39	Exp. Name: ***Write a C Program to store information using Structures with DMA***	Date: 2024-01-06

Aim:

Write a program to create a **list of nodes** using `self-referential structure` and print that data.

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

```
Enter an integer value :
```

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

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

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

```
Do u want another list (y|n) :
```

if the user gives the **input** as:

```
Do u want another list (y|n) : y
```

The input to the list is continued up to the user says `n` (No)

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

```
Enter an integer value : 20
Do u want another list (y|n) : y
Enter an integer value : 30
Do u want another list (y|n) : n
```

Finally, the program should print the result on the console as:

```
The elements in the single linked lists are : 10-->20-->30-->NULL
```

Note: Write the functions **create()** and **display()** in `CreateNodes.c`.

Source Code:

StructuresWithDma.c

```
#include <stdio.h>
#include <stdlib.h>
struct list {
        int data;
        struct list *next;
};
#include "CreateNodes.c"
void main() {
        struct list *first = NULL;
        first = create(first);
        printf("The elements in the single linked lists are : ");
        display(first);
}
```

CreateNodes.c

```
struct list* create(struct list *first) {
        char op;
        struct list *q, *temp;
        do {
                temp = (struct list *)malloc(sizeof(struct list)); // Allocate memory
                printf("Enter an integer value : ");
                scanf("%d",&temp->data ); // Read data
                temp -> next = NULL; // Place NULL
                if (first == NULL) {
                        first = temp ; // Assign temp to the first node
                } else {
                        q -> next = temp; // Create a link from the last node to new node
temp
                }
                q = temp;
                printf("Do u want another list (y|n) : ");
                scanf(" %c", &op);
        } while(op == 'y' || op == 'Y');
        return first;
}
void display(struct list *first) {
        struct list *temp = first;
        while ( temp!=NULL) { // Stop the loop where temp is NULL
                printf("%d-->", temp->data);
                temp = temp->next; // Assign next of temp to temp
        }
        printf("NULL\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter an integer value :
10
Do u want another list (y|n) :
y
Enter an integer value :
20
Do u want another list (y|n) :
y
Enter an integer value :
30
Do u want another list (y|n) :
n
The elements in the single linked lists are : 10-->20-->30-->NULL
```

S.No: 40	Exp. Name: ***Write a C program to demonstrate the differences between Structures and Unions***	**Date: 2024-01-18**

Aim:

Write a C program to demonstrate the differences between `structures` and `unions`.

The process is

6. Create a structure student-1 with members rollno, m1, m2, m3, total of int type and avg of float type
7. Read rollno, m1, m2 and m3 of student-1
8. Find and display total and average marks of student-1
9. Display the size of struct student-1
10. Create a union student-2 with members rollno, m1, m2, m3, total of int type and avg of float type
11. Read rollno, m1, m2 and m3 of student-2
12. Find and display total and average marks of student-2
13. Display the size of union student-2

Sample Input and Output:

```
Enter rollno and 3 subjects marks of student - 1 : 101 76 58 67
Total and average marks of student - 1 : 201 67.000000
Size of struct student - 1 : 24
Enter rollno of student - 2 : 102
Enter first subject marks of student - 2 : 76
Enter second subject marks of student - 2 : 87
Enter third subject marks of student - 2 : 69
Total marks of student - 2 : 232
Average marks of student - 2 : 77.333336
Size of union student - 2 : 4
```

Source Code:

```
StructureAndUnion.c
```

```
#include <stdio.h>
void main()
{
        struct student_1{
        int rollno,m1,m2,m3,total;
        float avg;
        }ss;
        union student_2{
        int rollno,m1,m2,m3,total;
        float avg;
        }us;
        int temp;
        printf("Enter rollno and 3 subjects marks of student - 1 : ");
        scanf("%d%d%d%d",&ss.rollno,&ss.m1,&ss.m2,&ss.m3);
        ss.total=ss.m1+ss.m2+ss.m3;
        ss.avg=ss.total/3.0;
        printf("Total and average marks of student - 1 : %d %f\n",ss.total,ss.avg);
        printf("Size of struct student - 1 : %lu\n",sizeof(ss));
        printf("Enter rollno of student - 2 : ");
        scanf("%d",&us.rollno);
        printf("Enter first subject marks of student - 2 : ");
        scanf("%d",&us.m1);
        temp=us.m1;
        printf("Enter second subject marks of student - 2 : ");
        scanf("%d",&us.m2);
        temp+=us.m2;
        printf("Enter third subject marks of student - 2 : ");
        scanf("%d",&us.m3);
        temp+=us.m3;
        us.total=temp;
        printf("Total marks of student - 2 : %d\n",us.total);
        us.avg=temp/3.0;
        printf("Average marks of student - 2 : %f\n",us.avg);
        printf("Size of union student - 2 : %lu\n",sizeof(us));
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter rollno and 3 subjects marks of student - 1 :
101 76 58 67
Total and average marks of student - 1 : 201 67.000000
Size of struct student - 1 : 24
Enter rollno of student - 2 :
102
Enter first subject marks of student - 2 :
76
Enter second subject marks of student - 2 :
87
Enter third subject marks of student - 2 :
```

```
69
Total marks of student - 2 : 232
Average marks of student - 2 : 77.333336
Size of union student - 2 : 4
```

Test Case - 2

User Output

```
Enter rollno and 3 subjects marks of student - 1 :
105 66 65 68
Total and average marks of student - 1 : 199 66.333336
Size of struct student - 1 : 24
Enter rollno of student - 2 :
106
Enter first subject marks of student - 2 :
88
Enter second subject marks of student - 2 :
89
Enter third subject marks of student - 2 :
79
Total marks of student - 2 : 256
Average marks of student - 2 : 85.333336
Size of union student - 2 : 4
```

Test Case - 3

User Output

```
Enter rollno and 3 subjects marks of student - 1 :
501 76 85 84
Total and average marks of student - 1 : 245 81.666664
Size of struct student - 1 : 24
Enter rollno of student - 2 :
502
Enter first subject marks of student - 2 :
99
Enter second subject marks of student - 2 :
57
Enter third subject marks of student - 2 :
69
Total marks of student - 2 : 225
Average marks of student - 2 : 75.000000
Size of union student - 2 : 4
```

Test Case - 4

User Output

```
Enter rollno and 3 subjects marks of student - 1 :
201 75 46 59
```

```
Total and average marks of student - 1 : 180 60.000000
Size of struct student - 1 : 24
Enter rollno of student - 2 :
201
Enter first subject marks of student - 2 :
66
Enter second subject marks of student - 2 :
57
Enter third subject marks of student - 2 :
61
Total marks of student - 2 : 184
Average marks of student - 2 : 61.333332
Size of union student - 2 : 4
```

S.No: 41	Exp. Name: ***Demonstrate left shift operation***	Date: 2024-01-06

Aim:
Write a C program to demonstrate left shift operation
Source Code:

shift.c

```
#include<stdio.h>
#include<string.h>
char val[10]="0000";;
void conv(int n)
{
if(n>1)
        conv(n/2);
        printf("%d",n%2);
}
void main()
{
        int n,d;
        printf("Enter an integer: ");
        scanf("%d",&n);
        printf("Original value: ");
        conv(n);
        printf("\nnumber of bits to left shift: ");
        scanf("%d",&d);
        int shift=n<<d;
        printf("After left shift: %d\n",shift);
        printf("Binary representation:");
        conv(shift);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter an integer:
12
Original value: 1100
number of bits to left shift:
2
After left shift: 48
Binary representation:110000
```

Test Case - 2

User Output

```
Enter an integer:
5
```

```
Original value: 101
number of bits to left shift:
3
After left shift: 40
Binary representation:101000
```

S.No: 42	Exp. Name: ***Copy the contents of one structure variable to another structure variable***	Date: 2024-01-18

Aim:
Write a C program to Copy the contents of one structure variable to another structure variable.

Let us consider a structure student, containing name, age and height fields.

Declare two structure variables to the structure student, read the contents of one structure variable and copy the same to another structure variable, finally display the copied data.

Note: *Driver code is provided to you in the* ***CopyStructureMain.c*** *file. You need to fill the missing code in* ***CopyStructureFunctions.c***

Source Code:

`CopyStructureMain.c`

```
#include <stdio.h>
#include "CopyStructureFunctions.c"

void main() {
	struct student s1, s2;
	read(&s1);
	s2 = copyStructureVariable(s1, s2);
	display(s2);
}
```

`CopyStructureFunctions.c`

```
#include <stdio.h>
struct student {
	//write the code
	char name[20];
int age;
float height;
}s;

void read(struct student *p) {
	printf("Enter student name, age and height: ");
	// Write the code to take inputs to structure
	scanf("%s%d%f", p->name,&p->age,&p->height);
}

struct student copyStructureVariable(struct student s1, struct student s2) {
	strcpy(s2.name, s1.name);
	s2.age= s1.age;
	s2.height= s1.height;
	return s2;
}

void display(struct student s) {
	//write your code here to display the structure data
	printf("Student name: %s\n",s.name);
	printf("Age: %d\n",s.age);
	printf("Height: %f\n",s.height);

}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter student name, age and height:
Yamuna 19 5.2
Student name: Yamuna
Age: 19
Height: 5.200000
```

Test Case - 2

User Output

```
Enter student name, age and height:
Kohli 21 5.11
Student name: Kohli
Age: 21
Height: 5.110000
```

S.No: 43	Exp. Name: ***Write a C Program to find nCr using Factorial recursive function***	**Date: 2024-01-06**

Aim:

Draw the flowchart and write a recursive **C** function to find the factorial of a number, `n!`, defined by **fact(n) = 1**, if n = 0. Otherwise **fact(n) = n * fact(n-1)**.

Using this function, write a **C** program to compute the binomial coefficient n_{C_r}. Tabulate the results for different values of **n** and **r** with suitable messages.

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

```
Enter the values of n and r :
```

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

```
Enter the values of n and r : 4 2
```

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

```
The value of 4c2 = 6
```

If the input is given as `2` and `5` then the program should print the result as:

```
Enter valid input data
```

Note: Write the recursive function **factorial()** in `Lab14a.c`.

Source Code:

Lab14a.c

```
int factorial(int n)
{
	if(n==0)
		return 1;
	else return n*factorial(n-1);
}
```

Lab14.c

```
#include <stdio.h>
#include "Lab14a.c"
void main() {
	int n, r;
	printf("Enter the values of n and r : ");
	scanf("%d %d", &n, &r);
	if (n >= r)
		printf("The value of %dc%d = %d\n", n, r, factorial(n) / (factorial(r) *
factorial(n - r)));
	else
		printf("Enter valid input data\n");
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter the values of n and r :
10 4
The value of 10c4 = 210
```

Test Case - 2

User Output

```
Enter the values of n and r :
7 9
Enter valid input data
```

Test Case - 3

User Output

```
Enter the values of n and r :
5 2
The value of 5c2 = 10
```

S.No: 44	Exp. Name: ***Write a Program to find the Length of a String***	Date: 2024-01-06

Aim:

Write a **C** program to find the `length` of a given string.

Sample Input and Output - 1:

```
Enter the string : CodeTantra
Length of CodeTantra : 10
```

Source Code:

StrLength.c

```
#include <stdio.h>
#include "StrLength1.c"
void main() {
	char str[30];
	printf("Enter the string : ");
	scanf("%s", str);
	printf("Length of %s : %d\n", str, myStrLen(str));
}
```

StrLength1.c

```
int myStrLen(char *str)
{
	int i=0;
	while(str[i]!='\0')
		{
			i++;
		}
	return i;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter the string :
CodeTantra
Length of CodeTantra : 10
```

Test Case - 2

User Output

```
Enter the string :
IndoUsUk
```

```
Length of IndoUsUk : 8
```

Test Case - 3

User Output

```
Enter the string :
MalayalaM
Length of MalayalaM : 9
```

Test Case - 4

User Output

```
Enter the string :
Oh!MyGod
Length of Oh!MyGod : 8
```

S.No: 45	Exp. Name: ***Transpose using functions.***	**Date: 2024-01-06**

Aim:
Write a C program to print the transpose of a matrix using functions.

Input Format
- First Line: The user will input the number of rows for the matrix.
- Second Line: The user will input the number of columns for the matrix.
- Subsequent Lines: The user will input the matrix elements row by row.

Output Format
- First Line: The program will print the matrix in its original form.
- Second Line: The program will print the transpose of the matrix.

Source Code:

```
transpose.c
```

```
#include <stdio.h>
int rows=5, cols=5;
int rows,cols;
void readMatrix(int mat[rows][cols])
{
        printf("Elements:\n");
for(int i=0;i<rows;i++)
        for(int j=0;j<cols;j++)
                scanf("%d",&mat[i][j]);
}
void printMatrix(int mat[rows][cols])
{
        printf("Matrix:\n");
        for(int i=0;i<rows;i++)
                {
                        for(int j=0;j<cols;j++)
                                printf("%d ",mat[i][j]);
                        printf("\n");
                }
}
void transposeMatrix(int mat[rows][cols])
{
        printf("Transpose:\n");
        for(int i=0;i<cols;i++)
                {
                        for(int j=0;j<rows;j++)
                                printf("%d ",mat[j][i]);
                        printf("\n");
                }
}




int main() {
    printf("rows: ");
    scanf("%d", &rows);
    printf("columns: ");
    scanf("%d", &cols);
    int matrix[rows][cols];

    // Input: Read the matrix elements
    readMatrix(matrix);

    // Print the original matrix
    printMatrix(matrix);

    // Print the transpose of the matrix
    transposeMatrix(matrix);

    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
rows:
2
columns:
2
Elements:
8 9
6 5
Matrix:
8 9
6 5
Transpose:
8 6
9 5
```

Test Case - 2

User Output

```
rows:
1
columns:
2
Elements:
6 9
Matrix:
6 9
Transpose:
6
9
```

S.No: 46	Exp. Name: ***Demonstrate numerical integration of differential equations using Euler's method***	**Date: 2024-01-18**

Aim:
Write a C function to demonstrate the numerical integration of differential equations using Euler's method.

Your program should prompt the user to input the initial value of ***y*** ***(y_o)***the initial value of ***t*** ***(t_o)*** the step size ***(h)***.and the end value for***t.*** Implement the Euler's method in a function, and print the values of***t*** and***y*** at each step.

The formula for Euler's Method:

y_{next}= y + h * f(y ,t)
where***f(y, t)*** is the derivative function representing ***dy/dt***in the given Ordinary differential equation.

Note: print the values of ***t*** and ***y*** up to 2 decimal places.

Source Code:

euler.c

```
#include <stdio.h>
void main()
{
        float y0,t0,h,t,x0;
        printf("initial value of y (y0): ");
        scanf("%f",&y0);
        printf("initial value of t (t0): ");
        scanf("%f",&t0);
        printf("step size (h): ");
        scanf("%f",&h);
        printf("end value for t: ");
        scanf("%f",&t);
        x0=t0;
        while(t0<t)
                {
                        printf("t = %.2f y = %.2f\n",t0,y0);
                        t0+=h;
                        y0=y0+h*(x0*y0);
                        x0=x0+h;
                }
}
/*/double f(double y, double t) {
void eulerIntegration(  ) {
int main() {
}*/
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
initial value of y (y0):
```

```
1
initial value of t (t0):
1
step size (h):
3
end value for t:
10
t = 1.00 y = 1.00
t = 4.00 y = 4.00
t = 7.00 y = 52.00
```

Test Case - 2

User Output

```
initial value of y (y0):
1
initial value of t (t0):
1
step size (h):
3
end value for t:
3
t = 1.00 y = 1.00
```

S.No: 47	Exp. Name: ***Fibonacci series up to the given number of terms using Recursion***	Date: 2024-01-18

Aim:
Write a program to display the fibonacci series up to the given number of terms using recursion process.
Source Code:

fibonacciSeries.c

```
#include <stdio.h>
#include "fibonacciSeriesa.c"
void main() {
	int n, i;
	printf("n: ");
	scanf("%d", &n);
	printf("%d terms: ", n);
	for (i = 0; i < n; i++) {
		printf("%d ", fib(i));
	}
}
```

fibonacciSeriesa.c

```
// Complete the function fib()....
int fib(int i){
	int t1=0,t2=1,t3;
	if(i==0) return t1;
	else if (i==1)  return t2;
	else
		return fib(i-1) + fib(i-2);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
n:
4
4 terms: 0 1 1 2
```

Test Case - 2

User Output

```
n:
10
10 terms: 0 1 1 2 3 5 8 13 21 34
```

S.No: 49	Exp. Name: ***Write a C program to find the Factorial of a given number using Recursion***	Date: 2024-01-18

Aim:
Write a program to find the `factorial` of a given number using recursion process.

Note: Write the recursive function **factorial()** in `Program901a.c`.
Source Code:

Program901.c

```
#include <stdio.h>
#include "Program901a.c"
void main() {
	long int n;
	printf("Enter an integer : ");
	scanf("%ld", &n);
	printf("Factorial of %ld is : %ld\n", n ,factorial(n));
}
```

Program901a.c

```
long int factorial(long int n)
{
	if(n==0 || n==1)
	{
		return 1;
	}
	else
	{
		return n*factorial(n-1);
	}
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter an integer :
5
Factorial of 5 is : 120
```

Test Case - 2

User Output

```
Enter an integer :
4
Factorial of 4 is : 24
```

Test Case - 3

User Output

```
Enter an integer :
8
Factorial of 8 is : 40320
```

Test Case - 4

User Output

```
Enter an integer :
0
Factorial of 0 is : 1
```

S.No: 50	Exp. Name: ***Ackermann function using Recursion***	Date: 2024-01-18

Aim:

Write a program to implement Ackermann function using recursion process.

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

```
Enter two numbers :
```

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

```
Enter two numbers : 2 1
```

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

```
A(2, 1) = 5
```

Source Code:

AckermannFunction.c

```
#include <stdio.h>
#include "AckermannFunction1.c"
void main() {
        long long int m, n;
        printf("Enter two numbers : ");
        scanf("%lli %lli", &m, &n);
        printf("A(%lli, %lli) = %lli\n", m, n, ackermannFun(m, n));
}
```

AckermannFunction1.c

```
long long int ackermannFun(long long int m,long long int n)
{
        if (m==0)
        {
                return n+1;
        }
        else if (n == 0)
        {
                return ackermannFun(m-1,1);
                        }
        else{
                return ackermannFun(m-1,ackermannFun(m,n-1));
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter two numbers :
0 1
A(0, 1) = 2
```

Test Case - 2

User Output

```
Enter two numbers :
2 2
A(2, 2) = 7
```

Test Case - 3

User Output

```
Enter two numbers :
2 1
A(2, 1) = 5
```

Test Case - 4

User Output

```
Enter two numbers :
1 1
A(1, 1) = 3
```

Test Case - 5

User Output

```
Enter two numbers :
1 0
A(1, 0) = 2
```

Test Case - 6

User Output

```
Enter two numbers :
2 3
A(2, 3) = 9
```

S.No: 51	Exp. Name: ***Problem solving***	Date: 2024-01-18

Aim:

Write a program to find the `sum` of `n` natural numbers using recursion process.

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

```
Enter value of n :
```

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

```
Enter value of n : 6
```

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

```
Sum of 6 natural numbers = 21
```

Note: Write the recursive function **sum()** in `Program903a.c`.

Source Code:

Program903.c

```
#include <stdio.h>
#include "Program903a.c"
void main() {
	int n;
	printf("Enter value of n : ");
	scanf("%d", &n);
	printf("Sum of %d natural numbers = %d\n", n, sum(n));
}
```

Program903a.c

```
int sum(int n)
{
	if (n==1) return 1;
	else return n+sum(n-1);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter value of n :
5
Sum of 5 natural numbers = 15
```

Test Case - 2

User Output

```
Enter value of n :
9
Sum of 9 natural numbers = 45
```

S.No: 52	Exp. Name: ***Write a C program to Swap two values by using Call-by-Address method***	**Date: 2024-01-18**

Aim:

Write a program to `swap` two values by using **call by address** method.

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

```
Enter two integer values :
```

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

```
Enter two integer values : 12 13
```

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

```
Before swapping in main : a = 12 b = 13
After swapping in swap : *p = 13 *q = 12
After swapping in main : a = 13 b = 12
```

Note: Write the function **swap()** in `Program1002a.c` and do use the **printf()** function with a **newline** character (`\n`).

Source Code:

Program1002.c

```
#include <stdio.h>
#include "Program1002a.c"
void main() {
        int a, b;
        printf("Enter two integer values : ");
        scanf("%d %d", &a, &b);
        printf("Before swapping in main : a = %d b = %d\n", a, b);
        swap(&a, &b);
        printf("After swapping in main : a = %d b = %d\n", a, b);
}
```

Program1002a.c

```
void swap (int *p, int *q)
{
        int t;
        t=*p;
        *p=*q;
        *q=t;
        printf("After swapping in swap : *p = %d *q = %d\n",*p,*q);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter two integer values :
121 131
Before swapping in main : a = 121 b = 131
After swapping in swap : *p = 131 *q = 121
After swapping in main : a = 131 b = 121
```

Test Case - 2

User Output

```
Enter two integer values :
555 999
Before swapping in main : a = 555 b = 999
After swapping in swap : *p = 999 *q = 555
After swapping in main : a = 999 b = 555
```

Test Case - 3

User Output

```
Enter two integer values :
1001 101
Before swapping in main : a = 1001 b = 101
After swapping in swap : *p = 101 *q = 1001
After swapping in main : a = 101 b = 1001
```

Test Case - 4

User Output

```
Enter two integer values :
9999 2999
Before swapping in main : a = 9999 b = 2999
After swapping in swap : *p = 2999 *q = 9999
After swapping in main : a = 2999 b = 9999
```

Test Case - 5

User Output

```
Enter two integer values :
10101 11010
Before swapping in main : a = 10101 b = 11010
After swapping in swap : *p = 11010 *q = 10101
After swapping in main : a = 11010 b = 10101
```