S.No: 1	Exp. Name: ***Display hello world message***	**Date: 2025-02-24**

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

hello.c

```
#include <stdio.h>

int main(){
	printf("Hello World\n");
	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: 2025-03-10**

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>

int 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:\n");

        printf("Integer: %d\n",a);
        printf("Float: %f\n",b);
        printf("Character: %c\n",c);
        printf("Double: %lf\n",d);

        return 0;
}
```

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: 2025-03-03**

Aim:
Write a C program to perform arithmetic operations like **+, -, *, /,** and **%** on two input variables $num1$ and $num2$.

Input Format:
- The first line of input is an integer representing the value for the first number, $num1$.
- The second line of input is an integer representing the value of the second number, $num2$.

Output Format:
- The program prints the integers that represent the results of addition, subtraction, multiplication, division, and modulus, each on a new line.

Note:
- For Division and Modulo operations, the value of $num2$. must be greater than 0.
- Add a new line character (\n) at the end of the output.

Source Code:

arithmeticOperations.c

```
// Type Content here...
#include <stdio.h>
int main(){
	int a,b;
	scanf("%d %d",&a,&b);
	printf("%d\n",a+b);
	printf("%d\n",a-b);
	printf("%d\n",a*b);
	printf("%d\n",a/b);
	printf("%d\n",a%b);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
9
```

8
17
1
72
1
1

Test Case - 2
User Output
1000
2
1002
998
2000
500
0

S.No: 4	Exp. Name: ***Temperature conversions from Centigrade to Fahrenheit and vice versa.***	**Date: 2025-03-03**

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

Input Format:
- Input a single floating-point number representing the temperature in Celsius to be converted to Fahrenheit.

Output Format:
- The program will print the temperature in both Celsius and Fahrenheit, formatted to two decimal places.

Note: Refer to the visible test cases for print statements.
Source Code:

temperature.c

```
// Type Content here...
#include <stdio.h>
int main(){
        double a,b;
        scanf("%lf",&a);
        b=(a*9/5)+32;
        printf("%.2lf Celsius = %.2lf Fahrenheit\n",a,b);
}
```

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

```
 -20.00 Celsius = -4.00 Fahrenheit
```

S.No: 5	Exp. Name: ***Pre and Post Increment Operations***	**Date: 2025-06-04**

Aim:
Write a C program to scan an input and perform pre and post increment operations on it and display the result.

Input format:
- Input prompts the user to enter an integer number.

Output format:
- The program outputs the original number, the value after post-increment, and the value after pre-increment.

Source Code:

prePostIncr.c

```
#include <stdio.h>
int main() {
    int number;
    // Input
    printf("Integer: ");
    //write the missing code
        scanf("%d",&number);

    // Display the results
    printf("Original Number: %d\n",  number);
    printf("After Post-increment: %d\n", number++ );
    printf("After Pre-increment: %d\n",  ++number );
    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Integer:
9
Original Number: 9
After Post-increment: 9
After Pre-increment: 11
```

Test Case - 2
User Output
`Integer:`
12
`Original Number: 12`
`After Post-increment: 12`
`After Pre-increment: 14`

S.No: 6	Exp. Name: ***Perform all bit wise operations***	**Date: 2025-03-10**

Aim:
Write a C program to perform all bit wise operations.

Sample Input and Output :

```
5
3
AND: 1
OR: 7
XOR: 6
NOT a: -6
Left shift a by 1: 10
Right shift a by 1: 2
```

Note: perform left and right shift bitwise operations by 1 bit

Source Code:

bitwise.c

```
#include <stdio.h>

int main() {
        int a,b;
        scanf("%d",&a);
        scanf("%d", &b);
        printf("AND: %d\n",a&b);
        printf("OR: %d\n",a|b);
        printf("XOR: %d\n",a^b);
        printf("NOT a: %d\n",~a);
        // printf(" Not ~: ~%d = %d\n",b,~b);
        printf("Left shift a by 1: %d\n",a<<1);
        printf("Right shift a by 1: %d\n",a>>1);
        //rintf("Left << : %d << 1 = %d\n",b,b<<1);
        //printf("Right >> : %d >> 1 = %d\n",b,b>>1);

    return 0;



}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
12
6
AND: 4
OR: 14
XOR: 10
NOT a: -13
Left shift a by 1: 24
Right shift a by 1: 6
```

Test Case - 2

User Output

```
15
7
AND: 7
OR: 15
XOR: 8
NOT a: -16
Left shift a by 1: 30
Right shift a by 1: 7
```

Test Case - 3

User Output

```
36
24
AND: 0
OR: 60
XOR: 60
NOT a: -37
Left shift a by 1: 72
Right shift a by 1: 18
```

S.No: 7	Exp. Name: ***Extract the last two digits of a given integer n***	**Date: 2025-04-07**

Aim:
Write a C program to extract the last two digits of a given integer n, where the number of digits should be greater than 2.

Input Format:

- The input consists of a single integer n, where $n > 99$.

Output Format:

- If the integer n has more than two digits, output the last two digits in the following format:

```
Last two digits: <XY>
```

where **<XY>** represents the last two digits of the integer n.

- If the integer n has two digits or fewer, output the following error message:

```
The input taken should be greater than 99
```

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

Source Code:

extract.c

```
#include <stdio.h>


int main() {
	int a,b=0;
	printf("Integer: ");
	scanf("%d",&a);
	if(a<100){
		printf("The input taken should be greater than 
%d\n",a);
	}else{
	b+= a%100;
	printf("Last two digits: %.2d\n",b);
}

	return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Integer:
1000
Last two digits: 00
```

Test Case - 2

User Output

```
Integer:
456987
Last two digits: 87
```

Test Case - 3

User Output

```
Integer:
99
The input taken should be greater than 99
```

S.No: 8	Exp. Name: ***Greatest of three numbers using a conditional operator***	**Date: 2025-04-07**

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

Input Format:
- The first three lines of input each contain one integer representing $num1$, $num2$, and $num3$, respectively.

Output Format:
- The output displays the greatest of the three entered integers.

Source Code:

greatest.c

```
#include <stdio.h>

int main() {
    int num1, num2, num3, greatest;

    // Take input from user
    printf("num1: ");
        scanf("%d",&num1);


        printf("num2: ");
        scanf("%d",&num2);


        printf("num3: ");
        scanf("%d",&num3);


    // Find the greatest number using the conditional operator
    greatest = num1>num2?(num1>num3?num1:num3):(num2>num3?num2:num3);


    // Display the greatest number
    printf("Greatest number: %d\n",greatest);

    return 0;
}
```

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: 9	Exp. Name: ***Program to Swap Two Numbers Without Using a Third Variable***	**Date: 2025-04-07**

Aim:
Write a program to swap two integer values a and b without using a third variable

Input Format:
- Two integers separated by space, representing the values of$a$ and b

Output Format:
- A line showing the initial values a and b before swapping seperated by a space
- A line showing the values of$a$ and b after swapping without using third variable

Example:
Input:
```
23 67
```
Output:
```
a = 23 b = 67         //Before Swapping
a = 67 b = 23        //After Swapping
```

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

Source Code:

Program311.c

```
#include <stdio.h>
void main() {
        int a, b;
        scanf("%d %d", &a, &b);
        printf("a = %d b = %d\n", a, b);
        a = a+b;
        b = a-b;
        a = a-b;
        printf("a = %d b = %d\n", a, b);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output

```
23 67
a = 23 b = 67
a = 67 b = 23
```

Test Case - 2

User Output

```
99 89
a = 99 b = 89
a = 89 b = 99
```

Test Case - 3

User Output

```
77 78
a = 77 b = 78
a = 78 b = 77
```

S.No: 10	Exp. Name: ***Check whether a given input integer is in between two values x and y.***	**Date: 2025-04-07**

Aim:

Write a C program to check whether a given integer is in between two specified values, x (lower bound) and y (upper bound), inclusive.

Input Format:

- The first line contains an integer x — the lower bound.
- The second line contains an integer y — the upper bound.
- The third line contains an integer num — the number to check.

Output Format:

- If num is between x and y (inclusive), print:

`num is in between x and y`

- Otherwise, print:

`num is not in between x and y`

Replace num, x, and y with their actual values in the output.

Source Code:

between.c

```
#include <stdio.h>

int main(){
	int x,y,num;
	printf("lower bound(x): ");
	scanf("%d",&x);
	printf("upper bound(y): ");
	scanf("%d",&y);
	printf("number to check: ");
	scanf("%d",&num);
	if(num > x && num < y){
		printf("%d is in between %d and %d",num,x,y);

	}else{
		printf("%d is not in between %d and %d",num,x,y);
	}
	return 0;
}
```

Test Case - 1

User Output

```
lower bound(x):
8
upper bound(y):
100
number to check:
16
16 is in between 8 and 100
```

Test Case - 2

User Output

```
lower bound(x):
1
upper bound(y):
15
number to check:
30
30 is not in between 1 and 15
```

S.No: 11	Exp. Name: ***Check whether a given character is a vowel or a consonant or a digit or a special symbol***	**Date: 2025-04-07**

Aim:
Write a C program to check whether a given character is a vowel or a consonant or a digit or a special symbol.

Source Code:

check.c

```
#include <stdio.h>

int main(){
        char val;
        printf("Input: ");
        scanf("%c",&val);
        if(val == 'A' || val == 'E' || val =='I' || val =='O'|| val==
'U'|| val =='a'|| val=='e'|| val == 'i'|| val=='o'|| val=='u' ){
                printf("vowel\n");
        }else if((val>='A' && val<='Z') || (val >='a' && val<='z')){
                printf("consonant\n");
        }
else if(val>'0' && val<='9'){
        printf("digit\n");

}else{
        printf("special symbol\n");
}


        return 0;






}
```

Execution Results - **All test cases have succeeded!**

Test Case - 1

User Output

```
Input:
9
digit
```

Test Case - 2

User Output

```
Input:
?
special symbol
```

Test Case - 3

User Output

```
Input:
C
consonant
```

Test Case - 4

User Output

```
Input:
e
vowel
```

S.No: 12	Exp. Name: ***C program to find roots and nature of quadratic equation.***	**Date: 2025-04-21**

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

Source Code:

quad_Equation.c

```
#include <stdio.h>
#include <math.h>

int main() {
	float a,b,c,real,img,d,r1,r2;
	printf("Enter coefficients a, b and c: ");
	scanf("%f %f %f",&a,&b,&c);
	d = b*b - 4*a*c;
	if(d>0){
		r1 = -b+sqrt(d)/(2*a);
		r2 = -b-sqrt(d)/(2*a);


		printf("root1 = %.2f and root2 = %.2f",r1,r2);

	}else if(d==0){
		r1 = -b/(2*a);
		r2 = -b/(2*a);
		printf("\nroot1 = %.2f and root2 = %.2f",r1,r2);
	}else{
		r1 = (-b/(2*a));
		r2 = sqrt(-d)/(2*a);
		printf("root1 = %.2f+%.2fi and root2 = %.2f-
%.2fi",r1,r2,r1,r2);
	}
	return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

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

Test Case - 2

User Output

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

S.No: 13	Exp. Name: ***Arithmetic operations using switch***	**Date: 2025-04-21**

Aim:
Write a C program to perform arithmetic operations using switch case

Input Format:
- The first line contains two numbersseparated by a space.
- The second line contains a single character representing the operator

Output Format:
- If a valid operation is performed, output the result formatted to two decimal places.
- If the operator is invalid, output "Invalid operator".

Note: Add a new line character (**\n**) at the end of the output.
Source Code:

```
arithmetic.c
```

```
#include <stdio.h>

int main () {
        float a,b;
        char c;
        printf("Enter two numbers: ");
        scanf("%f %f",&a,&b);
        printf("choose an operator (+, -, *, /): ");
        scanf(" %c",&c);
        switch(c){
                case '+' :
                printf("%.2f + %.2f = %.2f",a,b,a+b);
                break;
                case '-' :
                printf("%.2f - %.2f = %.2f",a,b,a-b);
                break;
                case '*' :
                printf("%.2f * %.2f = %.2f",a,b,a*b);
                break;
                case '/' :
                printf("%.2f / %.2f = %.2f",a,b,a/b);
                break;

                default:
                        {
                printf("Invalid operator");
                        }

        }
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter two numbers:
6 7
choose an operator (+, -, *, /):
+
6.00 + 7.00 = 13.00
```

Test Case - 2

User Output

```
Enter two numbers:
23 5
choose an operator (+, -, *, /):
&
Invalid operator
```

Test Case - 3

User Output

```
Enter two numbers:
67 98
choose an operator (+, -, *, /):
-
67.00 - 98.00 = -31.00
```

Test Case - 4

User Output

```
Enter two numbers:
0 87
choose an operator (+, -, *, /):
*
0.00 * 87.00 = 0.00
```

Test Case - 5

User Output

```
Enter two numbers:
56 98
choose an operator (+, -, *, /):
/
56.00 / 98.00 = 0.57
```

S.No: 14	Exp. Name: ***Convert upper case character to lowercase and vice versa***	**Date: 2025-04-21**

Aim:
Write a C program to convert upper case characters to lowercase and vice versa.
Source Code:

changeCharCase.c

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

int main() {
        char str[100];


        printf("Enter a string: ");
        scanf(" %s",str);
        int a = strlen(str);
        for(int i = 0; i<a; i++){
                if(str[i] > 'A' && str[i] < 'Z'){
                        str[i] = str[i] + 32;
                }else if(str[i] >= 'a' && str[i] < 'z'){
                        str[i] = str[i] - 32;
                }

                }

  printf("Result: %s\n",str);
  return 0;



}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter a string:
```

CodeTantra

```
Result: cODEtANTRA
```

Test Case - 2

User Output

```
Enter a string:
aabbccDDEEFFGG
Result: AABBCCddeeffgg
```

S.No: 15	Exp. Name: ***Print Odd Numbers Between Specified Range.***	**Date: 2025-04-28**

Aim:
Write a C program to print odd numbers between a specified range.

Input Format:
- The first line of input contains two integers $start$ and end, separated by a space.

Output Format:
- The program will print all odd numbers between $start$ and end(both are inclusive), separated by a space. If the end is smaller than the $start$, print "**Invalid range"**.

Source Code:

oddNumbers.c

```
#include <stdio.h>
int main() {
        int  i,a,b;
        printf("Enter the range: ");
        scanf("%d %d",&a,&b);

        for(i= a ; i <=b ; i++){
                if(i % 2!= 0){

                        printf("%d ",i);
                }


                }
        if(a  > b){
                printf("Invalid range");
        }





        return 0;


}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter the range:
1 9
1 3 5 7 9
```

Test Case - 2

User Output

```
Enter the range:
9 1
Invalid range
```

S.No: 16	Exp. Name: ***Factors of a given number and check whether it is a prime or not.***	**Date: 2025-06-04**

Aim:
Write a C program to display the factors of a given number and check whether it is a prime or not.

Source Code:

factorsAndIsPrime.c

```
#include<stdio.h>

int main(){
        int fact=0;
        int num;
        printf("Integer: ");
        scanf("%d",&num);

   printf("Factors: ");
        for(int i=0;i<=num;i++){
                if(num%i==0){
                                printf("%d ",i);
                        fact++;
                }
        }

                if(fact == 2){
                        printf("\nPrime number\n");

                }else{
                        printf("\nNot a prime number\n");
                }

        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Integer:

```
16
Factors: 1 2 4 8 16
Not a prime number
```

Test Case - 2

User Output

```
Integer:
13
Factors: 1 13
Prime number
```

S.No: 17	Exp. Name: ***Check whether the given integer is Armstrong or not and display the sum of individual digits raised to power of n.***	**Date: 2025-06-05**

Aim:

Write a C program to check whether the given integer is Armstrong or not and display the sum of individual digits of a given integer raised to the power of n.

Source Code:

```
SumAndArmstrong.c
```

```
#include <stdio.h>
#include <math.h>

//write your code...

int main()  {
        int n,power;
        int sum=0,original,rem,digits=0,amstrongsum=0;
        printf("Integer: ");
        scanf("%d",&n);
        printf("Power(n): ");
        scanf("%d",&power);
        original = n;
        while(n!=0){
                rem = n%10;
                sum +=pow(rem,power);
                n=n/10;
                digits++;

        }
        n= original;
        while(n!=0){
                rem = n%10;
                amstrongsum+=pow(rem,digits);
                n=n/10;
        }

        if(amstrongsum == original){
                printf("Armstrong number\n");

        }else{
                printf("Not an Armstrong number\n");
        }
        printf("Sum: %d\n",sum);
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Integer: 
```

```
153
Power(n):
2
Armstrong number
Sum: 35
```

Test Case - 2

User Output

```
Integer:
121
Power(n):
2
Not an Armstrong number
Sum: 6
```

S.No: 18	Exp. Name: ***Demonstrate the usage of unconditional control statements***	**Date: 2025-06-05**

Aim:

Write a C Program to demonstrate the usage of unconditional control statements by printing the square of numbers upto n.

Source Code:

```
unconditional.c
```

```
#include <stdio.h>
int main() {
    int choice, num, i;

    while(1){// Unconditional Goto statement

    printf("Choose an option:\n");
    printf("1.Print squares upto n\n");
    printf("2.Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);
    switch (choice) {
            case 1:
                    printf("n: ");
                    scanf("%d",&num);
             for(i=1;i<=num;i++){
                     printf("%d ",i*i);

             }
            printf("\n");
            break;
            case 2:
                    printf("Exited\n");
                    return 0;



            default:
            printf("Invalid choice\n");
            continue;


        }



    }

    return 0;
}
```

Execution Results - **All test cases have succeeded!**

Test Case - 1

User Output

```
Choose an option:
1.Print squares upto n
2.Exit
Enter your choice:
1
n:
9
1 4 9 16 25 36 49 64 81
Choose an option:
1.Print squares upto n
2.Exit
Enter your choice:
2
Exited
```

Test Case - 2

User Output

```
Choose an option:
1.Print squares upto n
2.Exit
Enter your choice:
15
Invalid choice
Choose an option:
1.Print squares upto n
2.Exit
Enter your choice:
1
n:
5
1 4 9 16 25
Choose an option:
1.Print squares upto n
2.Exit
Enter your choice:
2
```

Test Case - 3

User Output

```
Choose an option:
1.Print squares upto n
2.Exit
Enter your choice:
1
n:
1
1
Choose an option:
1.Print squares upto n
2.Exit
Enter your choice:
10
Invalid choice
Choose an option:
1.Print squares upto n
2.Exit
Enter your choice:
2
Exited
```

S.No: 19	Exp. Name: ***Pattern***	**Date: 2025-06-05**

Aim:
Write a C program to display the pattern by taking n value from the user.
Source Code:

pattern.c

```
#include <stdio.h>
int main(){
        int i,j,n;
        scanf("%d",&n);
         for(i=0;i<n;i++) {
                for(j=0;j<i*2;j++){
                        printf(" ");
                }
                for(j=n-i;j>0;j--){
                        printf("%d ",j);
                }
                printf("\n");
         }
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
5
5 4 3 2 1
  4 3 2 1
    3 2 1
      2 1
        1
```

Test Case - 2

User Output

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

S.No: 20	Exp. Name: ***Write a C program to demonstrate Functions without arguments and without return value***	**Date: 2025-04-28**

Aim:

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

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

The output is:

```
...***...
Hello! CodeTantra
...***...
```

Source Code:

FunctionCategories2.c

```
#include <stdio.h>
// Write the functions

int print() {
        printf("...***...\n");
        return 0;

}
int hello() {
        printf("Hello! CodeTantra\n");
        return 0;
}





void main() {
        print();
        hello();
        print();
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
`...***...`
`Hello! CodeTantra`
`...***...`

S.No: 21	Exp. Name: ***Fill in the missing code***	**Date: 2025-04-28**

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

Note: Take **pi** value as **3.14**
The below code is to find the `area of circle` using functions.
Source Code:

FunctionCategories4.c

```
#include <stdio.h>
void area_circle(float);
void main() {
	float radius;
	printf("Enter radius : ");
	scanf("%f", &radius);
	area_circle(radius);
}
void area_circle( float radius) { //Correct the code
	// Write the code to calculate the area of circle
	float area = 3.14 * radius * radius;
	printf("Area of circle = %f\n", area);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter radius : 
11.23
Area of circle = 395.994476
```

Test Case - 2

User Output

```
Enter radius : 
```

```
24.74
```

```
Area of circle = 1921.892212
```

S.No: 22	Exp. Name: ***Fill in the missing code***	**Date: 2025-04-28**

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

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

Source Code:

FunctionCategories7.c

```
#include <stdio.h>
int factorial(int);
void main() {
	int number;
	printf("Enter a number : ");
	scanf("%d", &number);
	printf("Factorial of a given number %d = %d\n", number,
factorial(number));
}
int factorial(int number) {
	int i, fact = 1;
	for (i =1;i<=number;i++) {
		fact *=i;
		}
	return fact;



	 // Write the return statement
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter a number :
3
Factorial of a given number 3 = 6
```

Test Case - 2

User Output

```
Enter a number :
6
Factorial of a given number 6 = 720
```

S.No: 23	Exp. Name: ***Write a C program to demonstrate Functions without arguments and with return value***	**Date: 2025-04-28**

Aim:

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

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

Write the function **prime()**.

Sample Input and Output:

```
Enter a number : 5
The given number is a prime number
```

Source Code:

```
FunctionCategories8.c
```

```
#include <stdio.h>
int prime();
void main() {
        if (prime() == 0) {
                printf("The given number is a prime number\n");
        } else {
                printf("The given number is not a prime number\n");
        }
}

int prime() {
        int count=0,n;
        printf("Enter a number : ");
        scanf("%d",&n);

        for(int i=1;i<=n;i++){
                if(n %i == 0){
                        count++;
                }


        }
        if(count ==2){
                return 0;
        }else{
                return 1;
        }

}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter a number : 
5
The given number is a prime number
```

Test Case - 2

User Output

```
Enter a number :
27
The given number is not a prime number
```

Test Case - 3

User Output

```
Enter a number :
121
The given number is not a prime number
```

Test Case - 4

User Output

```
Enter a number :
1
The given number is not a prime number
```

Test Case - 5

User Output

```
Enter a number :
117
The given number is not a prime number
```

Test Case - 6

User Output

```
Enter a number :
137
The given number is a prime number
```

S.No: 24	Exp. Name: ***Program to find the LCM of the given numbers***	**Date: 2025-04-28**

Aim:
Write a Program to find the LCM of the given numbers

Use the concept of functions to write the code
Source Code:

lcm.c

```
#include <stdio.h>

int lcm(int a, int b) {
        int lcm;
    // Write your code here
        int max = a>b?a:b;
        while(1){
                if(max%a ==0 && max%b == 0){
                        lcm =max;
                        break;
                }
                max++;
        };


        return lcm;



}


void main() {
    int a, b;
    printf("Enter two integer values: ");
    scanf("%d %d", &a, &b);
    printf("The lcm of two numbers %d and %d: %d\n", a, b, lcm(a, b));
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter two integer values:
3 2
The lcm of two numbers 3 and 2: 6
```

Test Case - 2

User Output

```
Enter two integer values:
12 144
The lcm of two numbers 12 and 144: 144
```

S.No: 25	Exp. Name: ***Demonstrate the usage of header file and functions***	**Date: 2025-05-28**

Aim:

Create a header file that contains the following prototype:
i. int factorial(int); // non-recursive function
ii. int factorial_rec(int); //Recursive function
iii. int prime(int);
Use the above functions in a C program by including the above header file.

Source Code:

```
primeAndFactorial.c
```

```
#include <stdio.h>
#include "declarations.h" // Include the header file

int main() {
    int num;

    // Input
    printf("Integer: ");
    scanf("%d", &num);

    // Calculate and display factorial using the non-recursive function
    int fact = factorial(num);
    printf("Factorial: %d\n", fact);

    // Calculate and display factorial using the recursive function
    int fact_rec = factorial_rec(num);
    printf("Factorial(recursive): %d\n", fact_rec);

    // Check if the number is prime
    if (prime(num)) {
        printf("Prime number\n");
    } else {
        printf("Not a prime number\n");
    }

    return 0;
}

// Function to calculate factorial (non-recursive)

        int factorial(int n){
                if(n<0){
                return -1;
                }
                 int fact =1;
                for(int i=1;i<=n;i++){
                        fact*=i;

                }



         return fact;
}

// Function to calculate factorial (recursive)
```

```
int factorial_rec(int n) {
        if(n<0){
                return -1;
        }
        if(n == 0 || n ==1){
                return 1;

        }
        return n*factorial_rec(n-1);

}

// Function to check if a number is prime
int prime(int n) {
        if(n<=1){
                return 0;

        }
        for(int i=2;i*i <=n;i++){
                if(n%i == 0){
                        return 0;
                }
        }


        return 1;
















    //write the function code here
}
```

declarations.h

```
// Function prototypes
int factorial(int n);
int factorial_rec(int n);
int prime(int n);
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Integer:
9
Factorial: 362880
Factorial(recursive): 362880
Not a prime number
```

Test Case - 2

User Output

```
Integer:
7
Factorial: 5040
Factorial(recursive): 5040
Prime number
```

S.No: 26	Exp. Name: ***Pascal triangle***	**Date: 2025-06-05**

Aim:

Write a C program to display Pascal's triangle using functions.

Source Code:

printPascalsTriangle.c

```
// Type Content here...

#include <stdio.h>


void pascaltriangle(int rows){
        for(int i=0;i<rows;i++) {
                int val =1;

                //  print spaces
                for(int space=0;space<rows-i;space++){
                        printf(" ");
                }
                for(int j=0;j<=i;j++){
                        printf("%d ",val);
                        val = val*(i-j)/(j+1);

                }
                printf("\n");
        }
}




int main() {
        int rows;
        printf("no of rows: ");
        scanf("%d",&rows);

        pascaltriangle(rows);
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
no of rows:
3
   1
  1 1
 1 2 1
```

Test Case - 2

User Output

```
no of rows:
4
    1
   1 1
  1 2 1
 1 3 3 1
```

S.No: 27	Exp. Name: ***Read n integer values into an array and display them***	**Date: 2025-05-31**

Aim:
Write a C program to read n integer values into an array and display them.

Input format:
- The first line of input should be the size of the array.
- The second line of input contains integers separated by spaces.

Output format:
- The output should display the array in the format: "**Array: <Array elements separated by space>**".

Source Code:

displayArray.c

```
#include <stdio.h>



int main() {
	int n;

	printf("");
	scanf("%d",&n);
	int arr[n];

	for(int i = 0;i<n;i++){
		scanf("%d",&arr[i]);
	}
	printf("Array: ");

	for(int i =0; i<n;i++){
		printf("%d ", arr[i]);
	}

	return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output
5
1 2 3 4 5
`Array: 1 2 3 4 5`

Test Case - 2
User Output
6
45 65 30 55 9 4
`Array: 45 65 30 55 9 4`

S.No: 28	Exp. Name: ***Display the count of positive, negative, even, and odd numbers and their sum***	**Date: 2025-05-31**

Aim:
Write a C program to count and display the number of positive, negative, even, and odd numbers in a given array of integers and also display their sum.

Source Code:

```
Integers.c
```

```
#include <stdio.h>

int main() {
        int n;

        printf("size: ");
        scanf("%d",&n);
        int arr[n];
        printf("Elements:");
        for(int i =0;i<n;i++){
                scanf("%d",&arr[i]);
        }
        int ps=0;
        int ns=0;
        int os=0;
        int es=0;
        int pn=0;
        int nn=0;
        int en=0;
        int on=0;



        for(int i=0;i<n;i++){
                int val =arr[i];
                if(val > 0){
                        pn++;
                        ps+=arr[i];
                }else{
                        nn++;
                        ns+=arr[i];

                }
                if(val % 2){
                        en++;
                        es+=arr[i];
                }else{
                        on++;
                        os+=arr[i];
                }
        }
        printf("Positive numbers: %d\n",pn);
        printf("Negative numbers: %d\n",nn);
        printf("Even numbers: %d\n",on);
        printf("Odd numbers: %d\n",en);
        printf("Sum of positive numbers: %d\n",ps);
        printf("Sum of negative numbers: %d\n",ns);
        printf("Sum of even numbers: %d\n",os);
        printf("Sum of odd numbers: %d\n",es);
        return 0;



}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
size:
5
Elements:
1 2 -3 4 -5
Positive numbers: 3
Negative numbers: 2
Even numbers: 2
Odd numbers: 3
Sum of positive numbers: 7
Sum of negative numbers: -8
Sum of even numbers: 6
Sum of odd numbers: -7
```

Test Case - 2

User Output

```
size:
6
Elements:
9 5 6 -4 -8 -10
Positive numbers: 3
Negative numbers: 3
Even numbers: 4
Odd numbers: 2
Sum of positive numbers: 20
Sum of even numbers: -16
Sum of odd numbers: 14
```

S.No: 29	Exp. Name: ***C program to find the largest and the smallest of some numbers given by the user***	**Date: 2025-05-31**

Aim:

Write the c program to find the largest and the smallest of some numbers given by the user.

Source Code:

SmallLarge.c

```
#include <stdio.h>


int main(){
	int n;
	printf("Enter the number of elements: ");
	scanf("%d",&n);
	int arr[n];

	for(int i=0;i<n;i++){
		printf("Enter a number: ");
		scanf("%d",&arr[i]);
	}

	int largest = 0;
	int smallest = arr[0];

	for(int i=0;i<n;i++){
		if(arr[i]> largest){
			largest = arr[i];
		}
		else if(arr[i] < smallest){
			smallest = arr[i];
		}

	}
	printf("Largest number: %d\n",largest);
	printf("Smallest number: %d\n",smallest);
	return 0;
}
```

Test Case - 1

User Output

```
Enter the number of elements:
5
Enter a number:
26
Enter a number:
25
Enter a number:
14
Enter a number:
283
Enter a number:
37
Largest number: 283
Smallest number: 14
```

Test Case - 2

User Output

```
Enter the number of elements:
4
Enter a number:
56
Enter a number:
21
Enter a number:
47
Enter a number:
560
Largest number: 560
Smallest number: 21
```

S.No: 30	Exp. Name: ***Write a C program to find the Addition of Two matrices using Functions***	**Date: 2025-06-05**

Aim:
Write a program to find the addition of two matrices using functions.

Note: Write the functions **read1()**, **display()** and **additionOfTwoMatrices()** in matricesAddition.c

Source Code:

readMatrices.c

```
#include <stdio.h>
#include "matricesAddition.c"
void main() {
	int a[10][10], b[10][10], m, n, p, q;
	printf("Enter the size of the first matrix : ");
	scanf("%d%d", &m, &n);
	read1(a, m, n);
	printf("Enter the size of the second matrix : ");
	scanf("%d%d", &p, &q);
	read1(b, p, q);
	printf("The first matrix is\n");
	display(a, m, n);
	printf("The second matrix is\n");
	display(b, p, q);
	if (m == p && n == q) {
		additionOfTwoMatrices(a, b, m, n);
	} else {
		printf("Addition is not possible\n");
	}
}
```

matricesAddition.c

```
void read1(int arr[10][10],int row,int col){
        printf("Enter %d elements : ",row*col);
        for(int i=0;i<row;i++){
                for(int j=0;j<col;j++){
                        scanf("%d",&arr[i][j]);
                }
        }

}

void display(int arr[10][10],int row,int col){

        for(int i=0;i<row;i++){
                for(int j=0;j<col;j++){
                        printf("%d ",arr[i][j]);
                }
                printf("\n");
        }

}

void additionOfTwoMatrices(int a[10][10],int b[10][10],int row,int col)
{
        int c[10][10];

        for(int i =0;i<row;i++){
                for(int j=0;j<col;j++){
                        c[i][j] = a[i][j] + b[i][j];
                                }
        }

        printf("The Addition Matrix is\n");
        display(c,row,col);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter the size of the first matrix :
2 3
Enter 6 elements :
```

```
1 2 3 4 5 6
Enter the size of the second matrix :
2 3
Enter 6 elements :
9 6 5 8 4 7
The first matrix is
1 2 3
4 5 6
The second matrix is
9 6 5
8 4 7
The Addition Matrix is
10 8 8
12 9 13
```

Test Case - 2

User Output

```
Enter the size of the first matrix :
2 2
Enter 4 elements :
45 15 35 62
Enter the size of the second matrix :
2 3
Enter 6 elements :
63 95 84 72 35 62
The first matrix is
45 15
35 62
The second matrix is
63 95 84
72 35 62
Addition is not possible
```

S.No: 31	Exp. Name: ***Write a C program to find the Multiplication of two matrices using Functions.***	**Date: 2025-06-05**

Aim:
Write a program to find the **multiplication of two matrices** using **functions**.

Note: Write the functions **read1()**, **display()** and **mulitiplicationOfTwoMatrices()** in matrixMul**.c**

Source Code:

```
readMatrices.c
```

```
#include <stdio.h>
#include "matrixMul.c"
void main() {
	int a[10][10], b[10][10], m, n, p, q;
	printf("Enter the size of the first matrix : ");
	scanf("%d%d", &m, &n);
	read1(a, m, n);
	printf("Enter the size of the second matrix : ");
	scanf("%d%d", &p, &q);
	read1(b, p, q);
	printf("The first matrix is\n");
	display(a, m, n);
	printf("The second matrix is\n");
	display(b, p, q);
	if (n == p) {
		multiplicationOfTwoMatrices(a, b, m, n, q);
	} else {
		printf("Multiplication is not possible\n");
	}
}
```

```
matrixMul.c
```

```
void read1(int arr[10][10],int row,int col){
        int i,j;
        printf("Enter %d elements : ",row*col);
        for(i=0;i<row;i++){
                for(j=0;j<col;j++){
                        scanf("%d",&arr[i][j]);
                }
        }
}
void display(int arr[][10],int row,int col){
        for(int i=0;i<row;i++){
                for(int j=0;j<col;j++){
                        printf("%d ",arr[i][j]);
                }
                printf("\n");
        }
}

void multiplicationOfTwoMatrices(int a[][10],int b[][10],int m,int
n,int q){
        int c[10][10];
        int i,j;
        printf("The resultant matrix is\n");

        for(i=0;i<m;i++){
                for(j=0;j<q;j++){
                        c[i][j]= 0;

                        for(int k=0;k<n;k++){


                                c[i][j]+=a[i][k]*b[k][j];
                        }
                }
        }
        display(c,m,q);



}
```

Execution Results - **All test cases have succeeded!**

Test Case - 1

User Output

```
Enter the size of the first matrix :
2 3
Enter 6 elements :
8 6 9 5 4 7
Enter the size of the second matrix :
3 2
Enter 6 elements :
6 5 9 8 7 2
The first matrix is
8 6 9
5 4 7
The second matrix is
6 5
9 8
7 2
The resultant matrix is
165 106
115 71
```

Test Case - 2

User Output

```
Enter the size of the first matrix :
2 3
Enter 6 elements :
65 95 85 45 25 35
Enter the size of the second matrix :
2 3
Enter 6 elements :
96 85 74 25 36 14
The first matrix is
65 95 85
45 25 35
The second matrix is
96 85 74
25 36 14
Multiplication is not possible
```

S.No: 32	Exp. Name: ***Transpose using functions.***	**Date: 2025-05-31**

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;
//write your code..
void readMatrix(int matrix[rows][cols]){
        printf("Elements:\n");
        for(int i=0;i<rows;i++){
                for(int j=0;j<cols;j++){
                        scanf("%d",&matrix[i][j]);
                }
        }
}

void printMatrix(int matrix[rows][cols]){
        printf("Matrix:\n");
        for(int i=0;i<rows;i++){
                for(int j=0;j<cols;j++){
                        printf("%d ",matrix[i][j]);
                }
                printf("\n");
        }
}
void transposeMatrix(int matrix[rows][cols]){
        printf("Transpose:\n");
        for(int i=0;i<cols;i++){
                for(int j=0;j<rows;j++){
                        printf("%d ",matrix[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: 33	Exp. Name: ***Print the index of the element if it is present else print "No"***	**Date: 2025-05-31**

Aim:

Write a C program to check whether a given integer exists in a list of numbers and print its index value if it is present, otherwise print “No”.

Source Code:

```
search.c
```

```
#include <stdio.h>

int main(){
        int n;
        printf("size: ");
        scanf("%d",&n);

        int arr[n];
        printf("numbers: ");

        for(int i=0;i<n;i++){
                scanf("%d",&arr[i]);
        }
        int k;
        printf("number to search: ");
        scanf("%d",&k);
        int flag=0;

        for(int i=0;i<n;i++){
                if(arr[i] == k){
                        printf("found at index:%d",i);
                        flag++;
                        break;

                }

        }
        if(flag==0){
                printf("No");
        }



        return 0;



}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
size:
5
numbers:
69 6 8 4 9
number to search:
10
No
```

Test Case - 2

User Output

```
size:
3
numbers:
9 6 8
number to search:
9
found at index:0
```

S.No: 34	Exp. Name: ***Delete all vowels in a given string and display the remaining string***	**Date: 2025-05-31**

Aim:

Write a C program to delete all vowels in a given string and display the remaining string.

Source Code:

```
deleteVowels.c
```

```
#include <stdio.h>
#include <string.h>
void removevowel(char str[]){
        char s[100];
        int j=0;

        for(int i=0;str[i] !='\0';i++){
                if(str[i]!='a'&& str[i]!='e' && str[i]!='i'&&
str[i]!='o'&& str[i]!='u'&& str[i]!='A'&& str[i]!='E'&& str[i]!='I'&&
str[i]!='O'&& str[i]!='U'){
                        s[j] = str[i];
                        j++;
                }
        }
        s[j] ='\0';
        printf("After removing vowels: %s\n",s);



}

int main() {
        char str[100];
        printf("String: ");
        fgets(str,sizeof(str),stdin);



        for(int i=0;str[i]!='\0';i++){
                if(str[i] == '\n'){
                        str[i]= '\0';
                        break;
                }

        }
        removevowel(str);

        return 0;



}
```

Execution Results - **All test cases have succeeded!**

Test Case - 1

User Output

```
String:
CodeTantra
After removing vowels: CdTntr
```

Test Case - 2

User Output

```
String:
Programming
After removing vowels: Prgrmmng
```

S.No: 35	Exp. Name: ***Check string is palindrome or not***	**Date: 2025-05-31**

Aim:

Write a C program to check whether a given string is Palindrome or not by passing parameters to Function & returning value.

Source Code:

```
stringPalindrome.c
```

```c
#include <stdio.h>



int main() {
	char s[100];
	printf("string: ");
	fgets(s,sizeof(s),stdin);
	int i,j=0;
	while(s[i]!='\0'){
		if(s[i] == '\n'){
			s[i]='\0';
			break;
		}
		i++;
	}
	int len =0;
	while(s[len] !='\0'){
		len++;
	}
	int isPalandrome=1;
	int st=0;
	int end = len-2;

	while(st<end){
		if(s[st] != s[end]){
			isPalandrome =0;
			break;

		}
		st++;
		end--;


	}

	if(isPalandrome){
		printf("palindrome\n");

	}else{
		printf("not a palindrome\n");
	}


}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
string:
mam
palindrome
```

Test Case - 2

User Output

```
string:
CodeTantra
not a palindrome
```

S.No: 36	Exp. Name: ***Read two integers as strings and displays their sum***	**Date: 2025-06-01**

Aim:
Write a C program that reads two integers as strings and displays their sum

Source Code:

integerSum.c

```
#include <stdio.h>

int stringToInt(char str[]){
int res =0;
int i=0;

while(str[i] !='\0'){
        res = res*10 + (str[i] -'0');
        i++;
}
   return res;

}


int main() {
    char num1_str[100], num2_str[100];
        int a,b;
        printf("num1: ");
        scanf("%s",num1_str);


        printf("num2: ");
        scanf("%s",num2_str);
a = stringToInt(num1_str);
b =stringToInt(num2_str);

        printf("Sum: %d",a+b);





    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
num1:
9
num2:
8
Sum: 17
```

Test Case - 2

User Output

```
num1:
16
num2:
9
Sum: 25
```

S.No: 37	Exp. Name: ***10 predefined string-handling functions***	**Date: 2025-06-01**

Aim:
Write a C program to demonstrate the usage of at least 10 predefined string-handling functions.

Instructions:
The functions that are to be used are as follows
1. strcat() - Concatenates two strings.
2. strlen() - Returns the length of a string.
3. strcpy() - Copies one string to another.
4. strcmp() - Compares two strings.
5. strncpy() - Copies a specified number of characters from one string to another.
6. strncat() - Concatenates a specified number of characters from one string to another.
7. strcasecmp() - Case-insensitive string comparison.
8. strncasecmp() - Case-insensitive comparison of a specified number of characters.
9. strspn() - Returns the length of the initial segment of a string containing only specified characters.
10. strtok() - Tokenizes a string based on a delimiter (Here the delimiter used is comma(,)).

Note:
- All operations are performed on the strings string1 and string2.
- The single argument functions are to be performed on string1.
- While copying the string, copy string1 only
- Refer to the displayed test cases for better understanding.

Source Code:

```
stringHandling.c
```

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

int main() {
    char str1[50];
    char str2[50];
    char str3[50];

    printf("string1: ");
    scanf("%s", str1);

    printf("string2: ");
    scanf("%s", str2);

// write your code here...
        strcat(str1,str2);
        printf("1. strcat(): %s\n",str1);
        int len = strlen(str1);
        printf("2. strlen(): %d\n",len);
        strcpy(str3,str1);
        printf("3. strcpy(): %s\n",str3);
        int cmp = strcmp(str1,str2);
        printf("4. strcmp(): %d\n",cmp);
        int n;
        printf("characters to be copied: ");
        scanf("%d",&n);
        strncpy(str3,str1,n);
        str3[n]='\0';
        printf("5. strncpy(): %s\n",str3);
        int m;
        printf("characters to be concatenated: ");
        scanf("%d",&m);
        strncat(str1,str2,m);
        printf("6. strncat(): %s\n",str1);
        int casecmp = strcasecmp(str1,str2);
        printf("7. strcasecmp(): %d\n",casecmp);
        int s;
        printf("characters to be compared: ");
        scanf("%d",&s);
        int ncasecmp = strncasecmp(str1,str2,s);
        printf("8. strncasecmp(): %d\n",ncasecmp);
        char a[50];
        printf("Enter initial segment: ");
        scanf("%s",a);
        printf("9. strspn(): %zu\n",strspn(str1,a));
        char tok[] = {',','\0'};
        printf("10. strtok(): %s\n",strtok(str1,tok));
        return 0;


}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
string1:
aabb
string2:
ccdd
1. strcat(): aabbccdd
2. strlen(): 8
3. strcpy(): aabbccdd
4. strcmp(): -2
characters to be copied:
1
5. strncpy(): a
characters to be concatenated:
2
6. strncat(): aabbccddcc
7. strcasecmp(): -2
characters to be compared:
2
8. strncasecmp(): -2
Enter initial segment:
a
9. strspn(): 2
10. strtok(): aabbccddcc
```

Test Case - 2

User Output

```
string1:
a,b,c,d
```

```
string2:
a,b
1. strcat(): a,b,c,da,b
2. strlen(): 10
3. strcpy(): a,b,c,da,b
4. strcmp(): 44
characters to be copied:
1
5. strncpy(): a
characters to be concatenated:
2
6. strncat(): a,b,c,da,ba,
7. strcasecmp(): 44
characters to be compared:
3
8. strncasecmp(): 0
Enter initial segment:
a
9. strspn(): 1
10. strtok(): a
```

S.No: 38	Exp. Name: ***User-defined string-handling functions.***	**Date: 2025-06-01**

Aim:

Write a C program that implements the following user-defined string-handling functions:

i. To find the length of the given string.
ii. To copy the contents of one string to another.
iii. To reverse the contents of a string.
iv. To compare two strings.
v. To concatenate two strings.

Source Code:

```
userDefined.c
```

```c
#include <stdio.h>

// Function to find the length of a string
int stringLength(const char *str) {

        int i=0;

        while(str[i]!='\0'){

                i++;



        }
        return i;




}
// Function to copy the contents of one string to another
void stringCopy(char *dest, const char *src) {
        int i=0;

        while(src[i]!='\0'){
                dest[i] = src[i];
                i++;
        }
        dest[i] ='\0';




}
// Function to reverse the contents of a string
void stringReverse(char *str) {
        int i=0;
        while(str[i]!='\0'){
                i++;
        }

int st =0;
int end = i-1;
        while(st<end){
                int temp = str[st];
                str[st] = str[end];
                str[end] = temp;
                st++;
                end--;

        }



}
// Function to compare two strings
int stringCompare(const char *str1, const char *str2) {
        int i=0;
        while(str1[i] !='\0' && str2[i]!='\0'){
                if(str1[i]!=str2[i]){
                        return str1[i]-str2[i];
                }

        }
        return str1[i]-str2[i];


}
// Function to concatenate two strings
void stringConcatenate(char *dest, const char *src) {
        int i=0;
        int j=0;
        while(dest[i]!='\0'){
                i++;

        }
        while(src[j]!='\0'){
                dest[i++] = src[j++];
        }
        dest[i] = '\0';



}

int main() {
    char str1[100], str2[50], result[150];

    printf("string1: ");
    scanf("%s", str1);
    printf("string2: ");
    scanf("%s", str2);
    int length = stringLength(str1);
    printf("Length of str1: %d\n", length);

    stringCopy(result, str1);
    printf("Copy of str1: %s\n", result);

    stringReverse(str2);
    printf("Reverse of str2: %s\n", str2);

    int cmp = stringCompare(str1, str2);
    if (cmp == 0) {
        printf("str1 and str2 are equal\n");
    } else if (cmp < 0) {
        printf("str1 is less than str2\n");
    } else {
        printf("str1 is greater than str2\n");
    }

    stringCopy(result, str1);
    stringReverse(str2);
    stringConcatenate(result, str2);
    printf("Concatenation of str1 and str2: %s\n", result);

    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
string1:
Code
string2:
Tantra
Length of str1: 4
Copy of str1: Code
Reverse of str2: artnaT
str1 is less than str2
Concatenation of str1 and str2: CodeTantra
```

Test Case - 2

User Output

```
string1:
Hello
string2:
World
Length of str1: 5
Copy of str1: Hello
Reverse of str2: dlroW
str1 is less than str2
Concatenation of str1 and str2: HelloWorld
```

S.No: 39	Exp. Name: ***Demonstrate the usage of pointers.***	**Date: 2025-06-04**

Aim:

Write a C program to demonstrate the usage of pointers.

1. Declare an integer variable **"number"** and an integer pointer **"pointer"**. Allow the user to input an integer, store it in **"number"**, and then use the pointer **"pointer"** to access and modify the value of **"number"**.
2. Declare an integer variable **"size"**, and an integer array **"numbers"** with a size specified by the user. Use a pointer **"arrPointer"** to input integers into the array. After input, use the same pointer to access and print the elements of the array.

Source Code:

```
usageOfPointers.c
```

```
#include <stdio.h>
int main() {
    int number;
    int *pointer; // Declared an integer pointer
        int modificationVal;
    printf("Enter number: ");
    scanf("%d", &number);

    // Assign the address of 'number' to 'pointer'

        pointer =&number;


    // Access and print the value using the pointer
        printf("number: %d\n",*pointer);



    // Take new value to modify the existing number
        printf("Enter modification value: ");
        scanf("%d",&modificationVal);




    // Modify the value of 'number' through the pointer and print it
    pointer = &modificationVal;
        printf("number: %d\n",*pointer);

    int size;
    printf("size: ");
    scanf("%d", &size);
    int numbers[size];
    int *arrPointer = numbers;
    printf("Elements: ");
    // Input integers into the numbers array from the user
        for(int i =0; i<size;i++) {
                scanf("%d",arrPointer+i);
        }


    printf("Array elements accessed using a pointer: ");
    arrPointer = numbers;
    // print the elements in the array using pointer variable
        for(int i=0;i<size;i++){
                        printf("%d ",arrPointer[i]);
                }

    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter number:
16
number: 16
Enter modification value:
9
number: 9
size:
5
Elements:
1 2 3 4 5
Array elements accessed using a pointer: 1 2 3 4 5
```

Test Case - 2

User Output

```
Enter number:
98
number: 98
Enter modification value:
10
number: 10
size:
3
Elements:
1 2 3
Array elements accessed using a pointer: 1 2 3
```

S.No: 40	Exp. Name: ***Demonstrate dynamic memory allocation functions to add n elements and display their average.***	**Date: 2025-06-04**

Aim:

Write a C program that uses dynamic memory allocation functions to add n elements and display their average.

Input Format

- First line: The program should print "**n:** " and accept an integer input representing the number of elements.
- Second line: The program should print "**Enter Elements:** " and accept n space-separated integer values.

Output Format:

- Print the average of the entered integers in the format: "**Average: <value>**" where **<value>** is printed with two decimal places.

Source Code:

```
dynamic.c
```

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int n;
    int *arr;
    double sum = 0;
    double average;
    // Input the number of elements from the user
    printf("n: ");
scanf("%d",&n);


    // Dynamically allocate memory for the array
        arr = (int *)malloc(n*sizeof(int));




        // Input the elements from the user
    printf("Enter Elements: ");

    // write the code for array input
        for(int i=0;i<n;i++){
                scanf("%d",&arr[i]);
                sum = sum + arr[i];
        }


    // calculate and display average
        printf("Average: ");
        average = sum / n;
        printf("%.2f",average);


    // Free the dynamically allocated memory
        free(arr);

    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
n:
5
Enter Elements:
31 32 33 35 36
Average: 33.40
```

Test Case - 2

User Output

```
n:
3
Enter Elements:
65 69 87
Average: 73.67
```

S.No: 41	Exp. Name: ***Pointer arithmetic***	**Date: 2025-06-04**

Aim:

Write a C program to access the array elements and print them in reverse order using pointer arithmetic.

Source Code:

```
pointerArithmetic.c
```

```
#include <stdio.h>
int main() {
    int n;
    printf("size: ");
    // Input the size of array, n, from user
        scanf("%d",&n);


        //Initialise the array of size n
        int arr[n];


        // Initialize a pointer to the start of array
        int *pointer;

        pointer = arr;


    printf("Elements: ");
    // Input integers into the array from the user
  for(int i=0;i<n;i++){
          scanf("%d",pointer+i);
  }



    printf("Array elements accessed using pointer arithmetic: ");
    // Print the array using pointer arithmetic
    for(int i=0;i<n;i++){
                printf("%d ",*(pointer+i));
        }

    // Reset the pointer to the start of the array
    pointer = arr;

    printf("\nArray elements accessed using negative index with pointer
arithmetic:");
    // Access array elements using negative index with pointer
arithmetic

    for(int i= n-1;i>=0;i--){
                printf("%d ",*(pointer+i));
        }
    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
size:
5
Elements:
6 9 8 1 2
Array elements accessed using pointer arithmetic: 6 9 8 1 2
Array elements accessed using negative index with pointer
arithmetic:2 1 8 9 6
```

Test Case - 2

User Output

```
size:
8
Elements:
4 5 9 8 7 6 5 4
Array elements accessed using pointer arithmetic: 4 5 9 8 7 6 5 4
Array elements accessed using negative index with pointer
arithmetic:4 5 6 7 8 9 5 4
```

S.No: 42	Exp. Name: ***C Program Swap Numbers in Cyclic Order Using Call by Reference***	**Date: 2025-06-04**

Aim:
C Program Swap three numbers in Cyclic Order Using Call by Reference

The cyclic swap order should be:
- a should get the value of c.
- b should get the value of a.
- c should get the value of b.

Input format:
A single line contains three integers a, b, and c.

Output format:
Print the numbers after swapping in cyclic order using call by reference.

Source Code:

`cycleSwap.c`

```
#include <stdio.h>

void cyclicSwap(int *a, int *b, int *c) {
    int temp;
        temp = *a;
        *a = *c;
        *c = *b;
        *b = temp;









}

int main() {
    int a, b, c;

    // Input three integers
    scanf("%d %d %d", &a, &b, &c);

    // Call the cyclicSwap function
    cyclicSwap(&a, &b, &c);

    // Output the swapped values
    printf("%d %d %d", a, b, c);

    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
1 2 3
3 1 2

Test Case - 2
User Output
14 15 65
`65 14 15`

Test Case - 3
User Output
7 5 1
`1 7 5`

Test Case - 4
User Output
2011 521 635
`635 2011 521`

S.No: 43	Exp. Name: ***Demonstrate Pointers to Pointers and Array of Pointers***	**Date: 2025-06-04**

Aim:

Write a C program to demonstrate the following

i. Pointers to Pointers

- Declare an integer variable **x** and two pointers **ptr1** and **ptr2**. Use **ptr1** and **ptr2** to display the value of **x** with single and double indirection, and allow the user to update the value of **x** through double indirection.

ii. Array of Pointers

- Declare an integer variable **n** and create an array of integer pointers named **arr.** Allow the user to input the size of the array and then input **n** integers to store in the array using dynamic memory allocation. Finally, print the elements of the array.

Source Code:

```
pointersArray.c
```

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int x;
    int updatedValue;
// i. Pointers to Pointers
        int *ptr1, **ptr2;
        printf("number: ");
        scanf("%d",&x);
        ptr1 = &x;
        ptr2 = &ptr1;



        printf("Value of x: %d\n", x);
    printf("Value pointed to by ptr1: %d\n",*ptr1);
    printf("Value pointed to by ptr2 (using double indirection):
%d\n",**ptr2);

// Changing the value of x through double indirection
        printf("Modification value: ");
    scanf("%d",&updatedValue);
        **ptr2 = updatedValue;


    printf("Updated value of x through double indirection: %d\n", x);

// ii. Array of Pointers
    int n;
    printf("size: ");
    scanf("%d", &n);
    int *arr[n]; // Array of integer pointers

int a[n];
        printf("Elements: ");
        for(int i=0;i<n;i++){
                scanf("%d",a+i);
        }
printf("Array: ");
        for(int i=0;i<n;i++){
                printf("%d ",a[i]);
        }

    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
number:
8
Value of x: 8
Value pointed to by ptr1: 8
Value pointed to by ptr2 (using double indirection): 8
Modification value:
16
Updated value of x through double indirection: 16
size:
5
Elements:
1 2 3 4 5
Array: 1 2 3 4 5
```

Test Case - 2

User Output

```
number:
50
Value of x: 50
Value pointed to by ptr1: 50
Value pointed to by ptr2 (using double indirection): 50
Modification value:
45
Updated value of x through double indirection: 45
size:
3
Elements:
4 5 6
Array: 4 5 6
```

S.No: 44	Exp. Name: ***Demonstrate pointer to array and pointers to functions***	**Date: 2025-06-04**

Aim:
Write a C program to demonstrate the following
i. Pointer to Array:

- Declare an integer variable **n** and a pointer **ptr3** to an integer array. Allow the user to input the size of the array and then input **n** integers to store in the array. Use the pointer **ptr3** to access and print the elements of the array.

ii. Pointers to Functions:

- Declare a pointer to a function named operation that can point to functions taking two integers and returning an integer. Create two functions, add and subtract, that perform addition and subtraction, respectively. Allow the user to input two integers, and then use the operation pointer to call either the add or subtract function based on user choice. Finally, display the result of the operation.

Source Code:

```
pointerToArray.c
```

```
#include <stdio.h>

// Function to add two numbers
int add(int a,int b){
        return (a+b);
}


// Function to subtract two numbers
int subtract(int a,int b){
        return (a-b);
}


int main() {
        int n;
        printf("size: ");
        scanf("%d",&n);


        // iii. Pointer to Array

        int *ptr3;




        // Declare an array of the user-defined size
        int a[n];
        printf("Elements: ");



    // Take array inputs

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

        ptr3 = a;





    printf("Accessing array elements using pointer:");
    for (int i = 0; i < n; i++) {
        *ptr3 = a[i];
        printf("%d ", (*ptr3) );
    }
printf("\n");
    // iv. Pointers to Functions
        printf("Enter two integers: ");
        int numA,numB;
        scanf("%d %d",&numA,&numB);


    // Declare a pointer to a function
        int (*operation)();
        int result;


    operation = add; // Assign the function 'add' to the pointer
    result = operation(numA, numB);
    printf("Using pointer to add: %d\n", result);

    operation = subtract; // Assign the function 'subtract' to the
pointer
    result = operation(numA, numB);
    printf("Using pointer to subtract: %d\n", result);

    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
size:
5
Elements:
1 2 3 4 5
Accessing array elements using pointer:1 2 3 4 5
Enter two integers:
16 9
Using pointer to add: 25
Using pointer to subtract: 7
```

Test Case - 2
User Output
`size:`
7
`Elements:`
1 6 9 87 45 20 25
`Accessing array elements using pointer:1 6 9 87 45 20 25`
`Enter two integers:`
26 98
`Using pointer to add: 124`
`Using pointer to subtract: -72`

S.No: 45	Exp. Name: ***Access and display the members of the structure***	**Date: 2025-06-04**

Aim:
Write a C program to access and display the members of the structure.
Source Code:

structures.c

```
#include <stdio.h>


struct Person {
    char name[50];
    int age;
    float salary;
    //write your code here..

};

int main() {

struct Person p;
        printf("Name: ");
        scanf("%s",p.name);

        printf("Age: ");
        scanf("%d",&p.age);

        printf("Salary: ");
        scanf("%f",&p.salary);
        //write your code here..

        printf("Displaying information about the person:\n");
        printf("Name: %s\n",p.name);
        printf("Age: %d\n",p.age);
        printf("Salary: %.2f\n",p.salary);

        return 0;


}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Name:
Jack
Age:
25
Salary:
50000
Displaying information about the person:
Name: Jack
Age: 25
Salary: 50000.00
```

Test Case - 2

User Output

```
Name:
William
Age:
28
Salary:
60000
Displaying information about the person:
Name: William
Age: 28
Salary: 60000.00
```

S.No: 46	Exp. Name: ***Demonstrate different ways to access the structure elements using pointers.***	**Date: 2025-06-04**

Aim:

Write a C program that demonstrates different ways to access the structure elements using pointers.

Source Code:

```
accessStructures.c
```

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

// Define a structure named 'Person' with name, age, salary as
variables
struct Person {
  char name[50];
  int age;
  float salary;




};

int main() {
    // Declare a structure variable of type 'Person'
        struct Person person1;


    // Input values for the structure members from the user
    printf("Name: ");
    scanf("%s", person1.name);
    printf("Age: ");
    scanf("%d", &person1.age);
    printf("Salary: ");
    scanf("%f", &person1.salary);

    // Declare a pointer ptr to a structure of type 'Person' and assign
the address of 'person1'
        struct Person *ptr;
        ptr = &person1;





    // Access structure elements using the pointer directly
    printf("Accessing structure elements using the pointer directly:");
    //Print Name,Age,Salary(up to 2 decimal points)

        printf("\nName: %s\n",(*ptr).name);
        printf("Age: %d\n",(*ptr).age);
        printf("Salary: %.2f",(*ptr).salary);
    // Access structure elements using the pointer with the '->'
operator
    printf("\nAccessing structure elements using the pointer with the
'->' operator:");
    //Print Name,Age,Salary(up to 2 decimal points)
        printf("\nName: %s\n",ptr->name);
        printf("Age: %d\n",ptr->age);
        printf("Salary: %.2f",ptr->salary);




    // Access structure elements using the pointer with the '(*ptr).'
notation
    printf("\nAccessing structure elements using the pointer with the
'(*ptr)' notation:");
    //Print Name,Age,Salary(up to 2 decimal points)
        printf("\nName: %s\n",ptr->name);
        printf("Age: %d\n",ptr->age);
        printf("Salary: %.2f\n",ptr->salary);

}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Name:
Jack
Age:
21
Salary:
50000
Accessing structure elements using the pointer directly:
Name: Jack
Age: 21
Salary: 50000.00
Accessing structure elements using the pointer with the '->'
operator:
Name: Jack
```

```
Age: 21
Salary: 50000.00
Accessing structure elements using the pointer with the '(*ptr)'
notation:
Name: Jack
Age: 21
Salary: 50000.00
```

Test Case - 2

User Output

```
Name:
William
Age:
25
Salary:
45000
Accessing structure elements using the pointer directly:
Name: William
Age: 25
Salary: 45000.00
Accessing structure elements using the pointer with the '->'
operator:
Name: William
Age: 25
Salary: 45000.00
Accessing structure elements using the pointer with the '(*ptr)'
notation:
Name: William
Age: 25
Salary: 45000.00
```

S.No: 47	Exp. Name: ***C program to read the contents of a file and display them on the output screen***	**Date: 2025-06-04**

Aim:
Write a C program to read the contents of a file and display them on the output screen.

Input Format:
- The program prompts the user to enter the name of the file they want to read.

Output Format:
- The program will display the contents of the specified file, printing each line as it reads them.

Note: Refer to visible test cases for better understanding.
Source Code:

```
readFile.c
```

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

int main() {
    // Declare a file pointer
    FILE *file;
    char filename[100];
    char line[1000];

    // Prompt the user for the file name
    printf("filename: ");
        scanf("%s",filename);



    // Open the file for reading
        file = fopen(filename,"r");




    // Read and display the contents of the file line by line
        printf("Contents of the file:\n");
        while(fgets(line,1000,file)){
                printf("%s",line);
        }
fclose(file);


    return 0;
}
```

input1.txt

```
CodeTantra
Start coding in 60 mins
```

input2.txt

```
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
Now is better than never.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
```

test1.txt

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Everything matters.
```

test2.txt

```
Hydrofoil is an underwater fin with a falt or curved wing-like
surface that is designed to lift a moving boat or ship
by means of the reaction upon its surface
```

test3.txt

```
Count the sentences in the file.
Count the words in the file.
Count the characters in the file.
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
filename:
input1.txt
Contents of the file:
CodeTantra
```

```
Start coding in 60 mins
```

Test Case - 2

User Output

```
filename:
input2.txt
Contents of the file:
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
Now is better than never.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
```

S.No: 48	Exp. Name: ***Copying a file to another file***	**Date: 2025-06-04**

Aim:
Write a C program to copy the data of file1 to file2 and display the data of file2

Input format:
- The input is a filename given by the user

Output format:
- The output should be the data in copied file

Source Code:

copyfile.c

```
#include <stdio.h>

int main() {
FILE *ptr;
char input[1000];
char line[100];
printf("file1:");
scanf("%s",input);
ptr = fopen(input,"r");
printf("Data from file2:\n");
while(fgets(line,1000,ptr)){
        printf("%s",line);
}

return 0;

}
```

input1.txt

```
CodeTantra
Start coding in 60 mins
```

input2.txt

```
Precision that helps you get in right everytime
```

test1.txt

```
Learn while you play story based development
```

test2.txt

```
Hydrofoil is an underwater fin with a flat or curved
wing like surface that is designed to lift
moving boat
```

test3.txt

```
Lets start coding
with CodeTantra
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
file1:
input1.txt
Data from file2:
CodeTantra
Start coding in 60 mins
```

Test Case - 2

User Output

```
file1:
input2.txt
Data from file2:
Precision that helps you get in right everytime
```

S.No: 49	Exp. Name: ***Write a C program to Count number of Characters, Words and Lines of a given File***	**Date: 2025-06-05**

Aim:

Write a program to `count` number of **characters**, **words** and **lines** of given text file.

- open a new file "`DemoTextFile2.txt`" in write mode
- write the content onto the file
- close the file
- open the same file in read mode
- read the text from file and find the characters, words and lines count
- print the counts of characters, words and lines
- close the file

Source Code:

```
Program1508.c
```

```
#include <stdio.h>
void main() {
	FILE *fp;
	char ch;
	int charCount= -1, wordCount= 1, lineCount= 1;
	fp = fopen("DemoTextFile2.txt", "w"); // Open a new file in
write mode
	printf("Enter the text with @ at end : ");
	while ((ch=getchar())!='@') {
		// Repeat loop till read @ at the end
		fputc(ch, fp); // Put read character onto the file
	}
	fputc('@', fp); // Put delimiter @ at the end on the file
	fclose(fp); // Close the file
	fp = fopen("DemoTextFile2.txt", "r"); // Open the existing file
in read mode
	do {
		ch= getc(fp);
		if (ch == ' '||ch=='\t' || ch=='\n' || ch=='\0'){
			wordCount++;

		}

		else{
			charCount++;
		}
		if (ch == '\n'){ // Write the condition to count lines
			lineCount++;}
	} while (ch!='@' && ch!=EOF); // Repeat loop till read @ at the
end
	fclose(fp);
	printf("Total characters : %d\n", charCount);
	printf("Total words : %d\n", wordCount);
	printf("Total lines : %d\n", lineCount);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter the text with @ at end :
```

Arise! Awake!

```
and stop not until
the goal is reached@
Total characters : 43
Total words : 10
Total lines : 3
```

Test Case - 2

User Output

```
Enter the text with @ at end :
Believe in your self
and the world will be
at your feet@
Total characters : 44
Total words : 12
Total lines : 3
```

S.No: 50	Exp. Name: ***Print the last n characters of a file by reading the file***	**Date: 2025-06-04**

Aim:
Write a C program to print the last **n** characters of a file by reading the file name and n value from the command line.

Source Code:

file.c

```
#include<stdio.h>
#include<stdlib.h>


int main(int argc,char *argv[]){
        FILE *file = fopen(argv[1],"r");
        int n = atoi(argv[2]);

        fseek(file,0,SEEK_END);
        long file_size =ftell(file);

        if(n > file_size){
                n = file_size;
        }
        fseek(file,-n,SEEK_END);

        for(int i=0;i<n;i++){
                putchar(fgetc(file));
        }
        fclose(file);
}
```

input1.txt

```
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
Now is better than never.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
```

input2.txt

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Everything matters.
```

test1.txt

```
CodeTantra
Start coding in 60 mins
```

test2.txt

```
Hydrofoil is an underwater fin with a falt or curved wing-like
surface that is designed to lift a moving boat or ship
by means of the reaction upon its surface
```

test3.txt

```
Count the sentences in the file.
Count the words in the file.
Count the characters in the file.
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
good idea.
```

Test Case - 2

User Output

```
verything matters.
```

S.No: 51	Exp. Name: ***Replace all the vowels in a given string with a given character***	**Date: 2025-06-05**

Aim:
Write a C program to replace all the vowels in a given string with a given character

Source Code:

replace.c

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

int main() {
	char c,s[100];
	printf("string: ");
	fgets(s,sizeof(s),stdin);
	printf("character to replace vowels with: ");
	scanf("%c",&c);
int n = strlen(s);
	for(int i=0;i<n;i++){

if(s[i]=='a'||s[i]=='e'||s[i]=='i'||s[i]=='o'||s[i]=='u'||s[i]=='A'||s[i]=='E'||s[i]=='I'||s[i]==
{
			s[i]=c;

		}
	}
	printf("New string: %s\n",s);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
string:
CodeTantra
character to replace vowels with:
T
New string: CTdTTTntrT
```

Test Case - 2
User Output
`string:`
programming
`character to replace vowels with:`
G
`New string: prGgrGmmGng`

S.No: 52	Exp. Name: ***Arithmetic operations using command-line arguments***	**Date: 2025-06-05**

Aim:
Write a C program to perform arithmetic operations using command-line arguments
Source Code:

arithmetic.c

```
#include <stdio.h>

#include<stdlib.h>

int main(int c,char*v[]){
        int a,b,ans;
        char op;
        a = atof(v[1]);
        b = atof(v[3]);

        op = v[2][0];
        switch(op){
                case'+':
                ans = a+b;
                break;
                case '-':
                ans =a-b;
                break;
                case '*':
                ans =a*b;
                break;
                case '/':
                ans = a/b;
                break;
        }
        printf("%d",ans);
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output
8

Test Case - 2
User Output
5

S.No: 53	Exp. Name: ***Reading and Writing Contents in File using Structures***	**Date: 2025-06-05**

Aim:

Write a C program that writes the contents to a file and reads the contents from a file using structures. Consider the filename as "person.txt".

Source Code:

```
writeAndRead.c
```

```
#include <stdio.h>
#include <stdlib.h>
// Define a structure to represent a person
struct Person {

    // Initialize name and age of a person
char name[50];
int age;


};
int main() {

        // Declare two struct variables for person1 and person2
        struct Person person1;
        struct Person person2;


    // Prompt the user to enter a name and store it in 'person1.age'
        printf("Enter name: ");
        scanf("%s",person1.name);


    // Prompt the user to enter a age and store it in 'person1.age'
        printf("Enter age: ");
        scanf("%d",&person1.age);


    // Open a file named "person.txt" in write mode.
    FILE *file = fopen("person.txt","w");
    fprintf(file, "%s %d\n",  person1.name,person1.age);

    // Close the file
        fclose(file);

    // Open the file "person.txt" in read mode
        file = fopen("person.txt","r");

    // Read the name and age from the file into 'person2'
        fscanf(file,"%s\n%d\n",person2.name,&person2.age);

    // Close the file
        fclose(file);

    // Display the person's details from 'person2'
        printf("Person details:\n");
        printf("Name: %s\n",person2.name);
        printf("Age: %d\n",person2.age);

    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter name:
Jack
Enter age:
25
Person details:
Name: Jack
Age: 25
```

Test Case - 2

User Output

```
Enter name:
William
Enter age:
28
Person details:
Name: William
Age: 28
```