**Week1:**

1.a Basic Linux environment and its editors like Vi,Vim & Emacs etc.

1.b Exposure to Turbo C,gcc.

1.c Explore to hacker rank or any other online coding platform and compiler environment.

1d)/\*Objective

In this challenge, we will learn some basic concepts of C that will get you started with the language. You will need to use the same syntax to read input and write output in many C challenges. As you work through these problems, review the code stubs to learn about reading from stdin and writing to stdout.

Task

This challenge requires you to print on a single line, and then print the already provided input string to stdout. If you are not familiar with C, you may want to read about the printf() command.

Example

The required output is:

Hello, World!

Life is beautiful

Function Description

Complete the main() function below.

The main() function has the following input:

string s: a string

Prints

\*two strings: \* "Hello, World!" on one line and the input string on the next line.

Input Format

There is one line of text, .

Sample Input 0

Welcome to C programming.

Sample Output 0

Hello, World!

Welcome to C programming.\*/

SOLUTIONS :

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main()

{

char s[100];

scanf("%[^\n]", &s);

printf("Hello, World!\n");

printf("%s",s);

return 0;

}

**Expected Output:-**

Input (stdin)

Welcome to C programming.

Expected Output

Hello, World!

Welcome to C programming.

1e) Write a simple program to read int, float, char and string using scanf() and display using printf() in all the above given platforms.

#include <stdio.h>

int main() {

int integer;

float floatingPoint;

char character;

char string[100];

// Reading values

printf("Enter an integer: ");

scanf("%d", &integer);

printf("Enter a floating-point number: ");

scanf("%f", &floatingPoint);

printf("Enter a character: ");

scanf(" %c", &character);

printf("Enter a string: ");

scanf(" %s", string);

// Displaying values

printf("\nInteger: %d\n", integer);

printf("Float: %f\n", floatingPoint);

printf("Character: %c\n", character);

printf("String: %s\n", string);

return 0;

}

**ExpectedOutput:-**

Enter an integer: 23

Enter a floating-point number: 423

Enter a character: f

Enter a string: handbs

Integer: 23

Float: 423.000000

Character: f

String: handbs

**Week2:**

**Hacker Rack Execise:-**

**Objective**

The fundamental data types in c are int, float and char. Today, we're discussing int and float data types.

The printf() function prints the given statement to the console. The syntax is printf("format string",argument\_list);. In the function, if we are using an integer, character, string or float as argument, then in the format string we have to write %d (integer), %c (character), %s (string), %f (float) respectively.

The scanf() function reads the input data from the console. The syntax is scanf("format string",argument\_list);. For ex: The scanf("%d",&number) statement reads integer number from the console and stores the given value in variable .

To input two integers separated by a space on a single line, the command is scanf("%d %d", &n, &m), where  and  are the two integers.

**Task**

Your task is to take two numbers of int data type, two numbers of float data type as input and output their sum:

1. Declare  variables: two of type int and two of type float.
2. Read  lines of input from stdin (according to the sequence given in the 'Input Format' section below) and initialize your  variables.
3. Use the  and  operator to perform the following operations:
   * Print the sum and difference of two int variable on a new line.
   * Print the sum and difference of two float variable rounded to one decimal place on a new line.

**Input Format**

The first line contains two integers.  
The second line contains two floating point numbers.

**Constraints**

* integer variables
* float variables

**Output Format**

Print the sum and difference of both integers separated by a space on the first line, and the sum and difference of both float (scaled to  decimal place) separated by a space on the second line.

**Sample Input**

10 4

4.0 2.0

**Sample Output**

14 6

6.0 2.0

**Explanation**

When we sum the integers  and , we get the integer . When we subtract the second number  from the first number , we get  as their difference.When we sum the floating-point numbers  and , we get . When we subtract the second number  from the first number , we get  as their difference.\*/

**Program:-**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main()

{

int a,b;

float c,d;

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

scanf("%f%f",&c,&d);

printf("%d %d \n",a+b,a-b);

printf("%f %f",c+d,c-d);

return 0;

}

**Excepted Output:-**

Input (stdin)

**10 4**

**4.0 2.0**

**Expected Output1:-**

**14 6**

**6.0 2.0**

Input (stdin)

**10 50**

**2.34 4.56**

Expected Output

**60 -40**

**6.9 -2.2**

2b)

**Objective**

This challenge will help you to learn how to take a character, a string and a sentence as input in C.To take a single character  as input, you can use scanf("%c", &ch ); and printf("%c", ch) writes a character specified by the argument char to stdout

char ch;

scanf("%c", &ch);

printf("%c", ch);

This piece of code prints the character .

You can take a string as input in C using scanf(“%s”, s). But, it accepts string only until it finds the first space.

In order to take a line as input, you can use scanf("%[^\n]%\*c", s); where  is defined as char s[MAX\_LEN] where  is the maximum size of . Here, [] is the scanset character. ^\n stands for taking input until a newline isn't encountered. Then, with this %\*c, it reads the newline character and here, the used \* indicates that this newline character is discarded.

**Note:** The statement: scanf("%[^\n]%\*c", s); will not work because the last statement will read a newline character, \n, from the previous line. This can be handled in a variety of ways. One way is to use scanf("\n"); before the last statement.

**Task**

You have to print the character, , in the first line. Then print  in next line. In the last line print the sentence, .

**Input Format**

First, take a character,  as input.  
Then take the string,  as input.  
Lastly, take the sentence  as input.

**Constraints**

Strings for  and  will have fewer than 100 characters, including the newline.

**Output Format**

Print three lines of output. The first line prints the character, .  
The second line prints the string, .  
The third line prints the sentence, .

**Sample Input 0**

C

Language

Welcome To C!!

**Sample Output 0**

C

Language

Welcome To C!!

SOLUTION:-

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main()

{

/\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/

char ch, s[100],sen[122];

scanf("%c",&ch);

scanf("%s\n",&s);

scanf("%[^\n]s",&sen);

printf("%c\n%s\n%s",ch,s,sen);

return 0;

}

OUTPUT:-

Input (stdin)

**C**

**Language**

**Welcome To C!!**

Your Output (stdout)

**C**

**Language**

**Welcome To C!!**

Expected Output

**C**

**Language**

**Welcome To C!!**

2C)/\* In this challenge, you will use logical bitwise operators. All data is stored in its binary representation. The logical operators, and C language, use  to represent true and  to represent false. The logical operators compare bits in two numbers and return true or false,  or , for each bit compared.

* Bitwise AND operator & The output of bitwise AND is *1* if the corresponding bits of two operands is *1*. If either bit of an operand is *0*, the result of corresponding bit is evaluated to *0*. It is denoted by &.
* Bitwise OR operator | The output of bitwise OR is *1* if at least one corresponding bit of two operands is *1*. It is denoted by |.
* Bitwise XOR (exclusive OR) operator ^ The result of bitwise XOR operator is *1* if the corresponding bits of two operands are opposite. It is denoted by .

For example, for integers 3 and 5,

3 = 00000011 (In Binary)

5 = 00000101 (In Binary)

AND operation OR operation XOR operation

00000011 00000011 00000011

& 00000101 | 00000101 ^ 00000101

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_

00000001 = 1 00000111 = 7 00000110 = 6

You will be given an integer , and a threshold, i1nnik$. Print the results of the and, or and exclusive or comparisons on separate lines, in that order.

**Example**  
The results of the comparisons are below:

a b and or xor

1 2 0 3 3

1 3 1 3 2

2 3 2 3 1

For the and comparison, the maximum is . For the or comparison, none of the values is less than , so the maximum is . For the xor comparison, the maximum value less than  is . The function should print:

2

0

2

**Function Description**

Complete the *calculate\_the\_maximum* function in the editor below.

*calculate\_the\_maximum* has the following parameters:

* *int n:* the highest number to consider
* *int k:* the result of a comparison must be lower than this number to be considered

**Prints**

Print the maximum values for the and, or and xor comparisons, each on a separate line.

**Input Format**

The only line contains  space-separated integers,  and .

**Constraints**

**Sample Input 0**

5 4

**Sample Output 0**

2

3

3

**Explanation 0**

**N=5, k=4**

**S={1,2,3,4,5}**

**All possible values of a and b are:**

* + The maximum possible value of  that is also  is , so we print  on first line.
  + The maximum possible value of  that is also  is , so we print  on second line.
  + The maximum possible value of  that is also  is , so we print  on third line.\*/

Program:-

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

void calculate\_the\_maximum(int n, int k) {

int max\_and = 0, max\_or = 0, max\_xor = 0;

for (int i = 1; i <= n; i++) {

for (int j = i + 1; j <= n; j++) {

int temp\_and = i & j;

int temp\_or = i | j;

int temp\_xor = i ^ j;

if (temp\_and > max\_and && temp\_and < k) {

max\_and = temp\_and;

}

if (temp\_or > max\_or && temp\_or < k) {

max\_or = temp\_or;

}

if (temp\_xor > max\_xor && temp\_xor < k) {

max\_xor = temp\_xor;

}

}

}

printf("%d\n%d\n%d", max\_and, max\_or, max\_xor);

}

int main() {

int n, k;

scanf("%d %d", &n, &k);

calculate\_the\_maximum(n, k);

return 0;

}

OUTPUT:-

Input (stdin)

**5 4**

Expected Output

**2**

**3**

**3**

Expected Output2:-

Input (stdin)

**5 5**

Output

**4**

**3**

**4**

2d) Conversion of Fahrenheit to Celsius and vice versa.

#include <stdio.h>

int main() {

int choice;

double temperature, converted\_temperature;

printf("Temperature Conversion Menu:\n");

printf("1. Fahrenheit to Celsius\n");

printf("2. Celsius to Fahrenheit\n");

printf("Enter your choice (1 or 2): ");

scanf("%d", &choice);

if (choice == 1) {

printf("Enter temperature in Fahrenheit: ");

scanf("%lf", &temperature);

converted\_temperature = (temperature - 32) \* 5 / 9;

printf("%.2lf°F is equal to %.2lf°C\n", temperature, converted\_temperature);

} else if (choice == 2) {

printf("Enter temperature in Celsius: ");

scanf("%lf", &temperature);

converted\_temperature = (temperature \* 9 / 5) + 32;

printf("%.2lf°C is equal to %.2lf°F\n", temperature, converted\_temperature);

} else {

printf("Invalid choice. Please enter 1 or 2.\n");

}

return 0;

}

Expected Output1:-

Temperature Conversion Menu:

1. Fahrenheit to Celsius

2. Celsius to Fahrenheit

Enter your choice (1 or 2): 1

Enter temperature in Fahrenheit: 48

48.00 F is equal to 8.89 C

**Expected Output2:-**

Temperature Conversion Menu:

1. Fahrenheit to Celsius

2. Celsius to Fahrenheit

Enter your choice (1 or 2): 2

Enter temperature in Celsius: 18

18.00░C is equal to 64.40░F

2e)Distance travelled by an object.

#include <stdio.h>

int main() {

double speed, time, distance;

// Input the speed in meters per second

printf("Enter the speed (m/s): ");

scanf("%lf", &speed);

// Input the time in seconds

printf("Enter the time (s): ");

scanf("%lf", &time);

// Calculate the distance

distance = speed \* time;

// Output the distance

printf("Distance traveled: %.2lf meters\n", distance);

return 0;

}

**Expected Output:-**

Enter the speed (m/s): 90

Enter the time (s): 120

Distance traveled: 10800.00 meters

**Expected Output2:-**

Enter the speed (m/s): 100

Enter the time (s): 60

Distance traveled: 6000.00 meters

2f)Calculate Simple interest and compound interest.

#include <stdio.h>

#include <math.h>

int main() {

double principal, rate, time, simple\_interest, compound\_interest;

// Input principal amount, interest rate, and time period

printf("Enter the principal amount: $");

scanf("%lf", &principal);

printf("Enter the annual interest rate (in percentage): ");

scanf("%lf", &rate);

printf("Enter the time period (in years): ");

scanf("%lf", &time);

// Calculate Simple Interest

simple\_interest = (principal \* rate \* time) / 100;

// Calculate Compound Interest

compound\_interest = principal \* (pow(1 + (rate / 100), time) - 1);

// Output the results

printf("\nSimple Interest = $%.2lf\n", simple\_interest);

printf("Compound Interest = $%.2lf\n", compound\_interest);

return 0;

}

**Expected Output1:-**

Enter the principal amount: $100000

Enter the annual interest rate (in percentage): 10

Enter the time period (in years): 3

Simple Interest = $30000.00

Compound Interest = $33100.00

**Expected Output 2:-**

Enter the principal amount: $200000

Enter the annual interest rate (in percentage): 12

Enter the time period (in years): 2

Simple Interest = $48000.00

Compound Interest = $50880.00

**Week3:**

3)a. Evaluate the following expressions

i. a/b\*c-b+a\*d/3

ii. j = (i++) + (++i)

#include <stdio.h>

int main() {

double a, b, c, d;

double result;

// Input values for a, b, c, and d

printf("Enter the value of a: ");

scanf("%lf", &a);

printf("Enter the value of b: ");

scanf("%lf", &b);

printf("Enter the value of c: ");

scanf("%lf", &c);

printf("Enter the value of d: ");

scanf("%lf", &d);

// Evaluate the expression

result = (a / b) \* c - b + (a \* d) / 3;

// Output the result

printf("Result: %.2lf\n", result);

return 0;

}

**Expected Output1:-**

Enter the value of a: 3

Enter the value of b: 4

Enter the value of c: 5

Enter the value of d: 4

Result: 3.75

**Expected Output2:-**

Enter the value of a: 2

Enter the value of b: 5

Enter the value of c: 6

Enter the value of d: 4

Result: 0.07

ii)

#include <stdio.h>

int main() {

int i = 5;

int j;

j = (i++) + (++i);

printf("i: %d\n", i);

printf("j: %d\n", j);

return 0;

}

OUTPUT:-

i: 7

j: 12

3b) Square root of a given number.

#include <stdio.h>

#include <math.h>

int main() {

double number, squareRoot;

// Input the number

printf("Enter a number: ");

scanf("%lf", &number);

// Calculate the square root

if (number >= 0) {

squareRoot = sqrt(number);

printf("Square root of %.2lf is %.2lf\n", number, squareRoot);

} else {

printf("Invalid input. Square root is not defined for negative numbers.\n");

}

return 0;

}

**ExpectedOutput1:-**

Enter a number: 64

Square root of 64.00 is 8.00

**Expected Output2:-**

Enter a number: 49

Square root of 49.00 is 7.00

3c) Find the area of circle, square, rectangle and triangle.

**Area of a Circle:**

#include <stdio.h>

#include <math.h>

int main() {

double radius, area;

printf("Enter the radius of the circle: ");

scanf("%lf", &radius);

if (radius >= 0) {

area = M\_PI \* pow(radius, 2);

printf("Area of the circle: %.2lf\n", area);

} else {

printf("Invalid input. Radius cannot be negative.\n");

}

return 0;

}

**ExpectedOtput1:-**

Enter the radius of the circle: 3

Area of the circle: 28.27

**ExpectedOutput2:-**

Enter the radius of the circle: 25

Area of the circle: 1963.50

**Area of the square:-**

#include <stdio.h>

int main() {

double side, area;

printf("Enter the side length of the square: ");

scanf("%lf", &side);

if (side >= 0) {

area = side \* side;

printf("Area of the square: %.2lf\n", area);

} else {

printf("Invalid input. Side length cannot be negative.\n");

}

return 0;

}

**ExpectedOutput1:-**

Enter the side length of the square: 5

Area of the square: 25.00

**ExpectedOutput2:-**

Enter the side length of the square: 25

Area of the square: 625.00

**Area of Rectangle:-**

#include <stdio.h>

int main() {

double length, width, area;

printf("Enter the length of the rectangle: ");

scanf("%lf", &length);

printf("Enter the width of the rectangle: ");

scanf("%lf", &width);

if (length >= 0 && width >= 0) {

area = length \* width;

printf("Area of the rectangle: %.2lf\n", area);

} else {

printf("Invalid input. Length and width cannot be negative.\n");

}

return 0;

}

**ExpectedOutput1:-**

Enter the length of the rectangle: 4

Enter the width of the rectangle: 5

Area of the rectangle: 20.00

**ExpectedOutput2:-**

Enter the length of the rectangle: 25

Enter the width of the rectangle: 45

Area of the rectangle: 1125.00

**Area of a Triangle:**

#include <stdio.h>

int main() {

double base, height, area;

printf("Enter the base of the triangle: ");

scanf("%lf", &base);

printf("Enter the height of the triangle: ");

scanf("%lf", &height);

if (base >= 0 && height >= 0) {

area = 0.5 \* base \* height;

printf("Area of the triangle: %.2lf\n", area);

} else {

printf("Invalid input. Base and height cannot be negative.\n");

}

return 0;

}

**ExpectedOutput1:-**

Enter the base of the triangle: 4

Enter the height of the triangle: 3

Area of the triangle: 6.00

**ExpectedOutput2:-**

Enter the base of the triangle: 34

Enter the height of the triangle: 12

Area of the triangle: 204.00

3d) Find the maximum of three numbers using conditional operator.

#include <stdio.h>

int main() {

int num1, num2, num3, max;

printf("Enter three numbers: ");

scanf("%d %d %d", &num1, &num2, &num3);

// Use the conditional operator to find the maximum

max = (num1 > num2) ? ((num1 > num3) ? num1 : num3) : ((num2 > num3) ? num2 : num3);

printf("The maximum of %d, %d, and %d is: %d\n", num1, num2, num3, max);

return 0;

}

**Expectedoutput1:-**

Enter three numbers: 5

4

3

The maximum of 5, 4, and 3 is: 5

**Expected Output2:-**

Enter three numbers: 8

7

6

The maximum of 8, 7, and 6 is: 8

3e)/\* Take marks of 5 subjects in integers, find the total in integer and average in float.\*/

#include <stdio.h>

int main() {

int marks[5]; // An array to store marks of 5 subjects

int total = 0; // Variable to store the total marks

float average; // Variable to store the average

// Input marks for 5 subjects

printf("Enter marks for 5 subjects:\n");

for (int i = 0; i < 5; i++) {

printf("Subject %d: ", i + 1);

scanf("%d", &marks[i]);

total += marks[i]; // Add the marks to the total

}

// Calculate the average

average = (float)total / 5; // Convert the total to a float for accurate average

// Output the total and average

printf("Total marks: %d\n", total);

printf("Average marks: %.2f\n", average);

return 0;

}

**ExpectedOutput1:**

Enter marks for 5 subjects:

Subject 1: 98

Subject 2: 68

Subject 3: 78

Subject 4: 87

Subject 5: 76

Total marks: 407

Average marks: 81.40

**ExpectedOutput2:-**

Enter marks for 5 subjects:

Subject 1: 67

Subject 2: 76

Subject 3: 56

Subject 4: 85

Subject 5: 56

Total marks: 340

Average marks: 68.00

**Week4:**

4A)

#include <assert.h>

#include <limits.h>

#include <math.h>

#include <stdbool.h>

#include <stddef.h>

#include <stdint.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char\* readline();

int main()

{

char\* n\_endptr;

char\* n\_str = readline();

int n = strtol(n\_str, &n\_endptr, 10);

if (n\_endptr == n\_str || \*n\_endptr != '\0') { exit(EXIT\_FAILURE); }

// Write Your Code Here

switch(n)

{

case 1:printf("one \n");

break;

case 2:printf("two \n");

break;

case 3:printf("three \n");

break;

case 4:printf("four \n");

break;

case 5:printf("five \n");

break;

case 6:printf("six\n");

break;

case 7:printf("seven\n");

break;

case 8:printf("eight \n");

break;

case 9:printf("nine \n");

break;

default :printf("Greater than 9");

break;

}

return 0;

}

char\* readline() {

size\_t alloc\_length = 1024;

size\_t data\_length = 0;

char\* data = malloc(alloc\_length);

while (true) {

char\* cursor = data + data\_length;

char\* line = fgets(cursor, alloc\_length - data\_length, stdin);

if (!line) { break; }

data\_length += strlen(cursor);

if (data\_length < alloc\_length - 1 || data[data\_length - 1] == '\n') { break; }

size\_t new\_length = alloc\_length << 1;

data = realloc(data, new\_length);

if (!data) { break; }

alloc\_length = new\_length;

}

if (data[data\_length - 1] == '\n') {

data[data\_length - 1] = '\0';

}

data = realloc(data, data\_length);

return data;

}

**ExpectedOuput1:-**

Input (stdin)

**8**

Expected Output

**Eight**

**ExpectedOutput2:-**

Input (stdin)

**10**

Expected Output

**Greater than 9**

**4B)** Roots of a Quadratic Equation.

#include <stdio.h>

#include <math.h>

int main() {

double a, b, c;

double discriminant, root1, root2;

printf("Enter the coefficients of the quadratic equation (a, b, and c): ");

scanf("%lf %lf %lf", &a, &b, &c);

// Calculate the discriminant

discriminant = (b \* b) - (4 \* a \* c);

// Check the value of the discriminant

if (discriminant > 0) {

// Two real and distinct roots

root1 = (-b + sqrt(discriminant)) / (2 \* a);

root2 = (-b - sqrt(discriminant)) / (2 \* a);

printf("Root 1: %.2lf\n", root1);

printf("Root 2: %.2lf\n", root2);

} else if (discriminant == 0) {

// One real root (repeated)

root1 = -b / (2 \* a);

printf("Root: %.2lf\n", root1);

} else {

// Complex roots

double realPart = -b / (2 \* a);

double imaginaryPart = sqrt(-discriminant) / (2 \* a);

printf("Root 1: %.2lf + %.2lfi\n", realPart, imaginaryPart);

printf("Root 2: %.2lf - %.2lfi\n", realPart, imaginaryPart);

}

return 0;

}

**ExpectedOtput1:-**

Enter the coefficients of the quadratic equation (a, b, and c): 8

6

4

Root 1: -0.38 + 0.60i

Root 2: -0.38 - 0.60i

**ExpectedOutput2:-**

Enter the coefficients of the quadratic equation (a, b, and c): 8

4

5

Root 1: -0.25 + 0.75i

Root 2: -0.25 - 0.75i

4C) Generate electricity bill.

#include <stdio.h>

int main() {

// Constants for billing rates

double unitRate = 0.15; // Cost per unit of electricity in dollars

double fixedCharge = 10.0; // Fixed monthly charge in dollars

// Customer information

char customerName[50];

int customerId;

int unitsConsumed;

// Input customer information

printf("Enter customer name: ");

fgets(customerName, sizeof(customerName), stdin);

printf("Enter customer ID: ");

scanf("%d", &customerId);

printf("Enter units consumed: ");

scanf("%d", &unitsConsumed);

// Calculate total cost

double totalCost = fixedCharge + (unitsConsumed \* unitRate);

// Generate the electricity bill

printf("\nElectricity Bill\n");

printf("Customer Name: %s", customerName);

printf("Customer ID: %d\n", customerId);

printf("Units Consumed: %d\n", unitsConsumed);

printf("Cost per Unit: $%.2lf\n", unitRate);

printf("Fixed Monthly Charge: $%.2lf\n", fixedCharge);

printf("Total Cost: $%.2lf\n", totalCost);

return 0;

}

**ExpectedOutput1:-**

Enter customer name: RAM

Enter customer ID: 34557568758

Enter units consumed: 234

Electricity Bill

Customer Name: RAM

Customer ID: 197830390

Units Consumed: 234

Cost per Unit: $0.15

Fixed Monthly Charge: $10.00

Total Cost: $45.10

**ExpectedOutput2:-**

Enter customer name: 637884

Enter customer ID: 32144

Enter units consumed: 30

Electricity Bill

Customer Name: 637884

Customer ID: 32144

Units Consumed: 30

Cost per Unit: $0.15

Fixed Monthly Charge: $10.00

Total Cost: $14.50

4D) /\*Simulate a calculator using switch case. \*/

#include <stdio.h>

int main() {

double num1, num2, result;

char op;

printf("Simple Calculator\n");

printf("Enter the first number: ");

scanf("%lf", &num1);

printf("Enter an operator (+, -, \*, /): ");

scanf(" %c", &op); // Note the space before %c to consume any leading whitespace

printf("Enter the second number: ");

scanf("%lf", &num2);

switch (op) {

case '+':

result = num1 + num2;

break;

case '-':

result = num1 - num2;

break;

case '\*':

result = num1 \* num2;

break;

case '/':

if (num2 != 0) {

result = num1 / num2;

} else {

printf("Error: Division by zero is not allowed.\n");

return 1; // Exit with an error code

}

break;

default:

printf("Error: Invalid operator.\n");

return 1; // Exit with an error code

}

printf("Result: %.2lf %c %.2lf = %.2lf\n", num1, op, num2, result);

return 0;

}

**ExpectedOutput1:-**

Simple Calculator

Enter the first number: 56

Enter an operator (+, -, \*, /): \*

Enter the second number: 45

Result: 56.00 \* 45.00 = 2520.00.

**ExpectedOutput2:-**

Simple Calculator

Enter the first number: 34

Enter an operator (+, -, \*, /): /

Enter the second number: 24

Result: 34.00 / 24.00 = 1.42

4E)/\* Find the given year is a leap year or not.\*/

#include <stdio.h>

int main() {

int year;

printf("Enter a year: ");

scanf("%d", &year);

if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {

printf("%d is a leap year.\n", year);

} else {

printf("%d is not a leap year.\n", year);

}

return 0;

}

**ExpectedOutput1:-**

Enter a year: 2024

2024 is a leap year.

**ExpectedOutput2:-**

Enter a year: 2034

2034 is not a leap year.

**Week5**: **Loops**

A)/\* **Objective**

In this challenge, you will learn the usage of the *for* loop, which is a programming language statement which allows code to be executed until a terminal condition is met. They can even repeat forever if the terminal condition is never met.

The syntax for the for loop is:

for ( <expression\_1> ; <expression\_2> ; <expression\_3> )

<statement>

* *expression\_1* is used for intializing variables which are generally used for controlling the terminating flag for the loop.
* *expression\_2* is used to check for the terminating condition. If this evaluates to false, then the loop is terminated.
* *expression\_3* is generally used to update the flags/variables.

The following loop initializes  to 0, tests that  is less than 10, and increments  at every iteration. It will execute 10 times.

for(int i = 0; i < 10; i++) {

...

}

**Task**

For each integer  in the interval  (given as input) :

* If , then print the English representation of it in lowercase. That is "one" for , "two" for , and so on.
* Else if  and it is an even number, then print "even".
* Else if  and it is an odd number, then print "odd".

**Input Format**

The first line contains an integer, .  
The seond line contains an integer, .

**Constraints**

**Output Format**

Print the appropriate English representation,even, or odd, based on the conditions described in the 'task' section.

**Note:**

**Sample Input**

8

11

**Sample Output**

eight

nine

even

odd

Change Theme

Language: C

More

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

**#include**·**<**stdio.h**>**

**#include**·**<**string.h**>**

**#include**·**<**math.h**>**

**#include**·**<**stdlib.h**>**

**int**·main()·{

····**int**·a,·b;

····scanf("%d\n%d",·&a,·&b);

····**for**·(**int**·i·=·a;·i·<=·b;·i++)·{

········**if**·(i·==·1)·{

············printf("one\n");

········}·**else**·**if**·(i·==·2)·{

············printf("two\n");

········}·**else**·**if**·(i·==·3)·{

············printf("three\n");

········}·**else**·**if**·(i·==·4)·{

············printf("four\n");

········}·**else**·**if**·(i·==·5)·{

············printf("five\n");

········}·**else**·**if**·(i·==·6)·{

············printf("six\n");

········}·**else**·**if**·(i·==·7)·{

············printf("seven\n");

········}·**else**·**if**·(i·==·8)·{

············printf("eight\n");

Line: 38 Col: 1

Submit Code

Run Code

Upload Code as File

**Test against custom input**

**Congratulations!**

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

**Sample Test case 0**

Input (stdin)

Download

* **8**
* **11**

Your Output (stdout)

* **eight**
* **nine**
* **even**
* **odd**

Expected Output

Download

* **eight**
* **nine**
* **even**
* **odd \*/**

**Program:-**

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main() {

int a, b;

scanf("%d\n%d", &a, &b);

for (int i = a; i <= b; i++) {

if (i == 1) {

printf("one\n");

} else if (i == 2) {

printf("two\n");

} else if (i == 3) {

printf("three\n");

} else if (i == 4) {

printf("four\n");

} else if (i == 5) {

printf("five\n");

} else if (i == 6) {

printf("six\n");

} else if (i == 7) {

printf("seven\n");

} else if (i == 8) {

printf("eight\n");

} else if (i == 9) {

printf("nine\n");

} else if (i % 2 == 0) {

printf("even\n");

} else {

printf("odd\n");

}

}

return 0;

}

Expected Output1:-

Expected Output2:-

5B) **Objective**

The modulo operator, %, returns the remainder of a division. For example, 4 % 3 = 1 and 12 % 10 = 2. The ordinary division operator, /, returns a truncated integer value when performed on integers. For example, 5 / 3 = 1. To get the last digit of a number in base 10, use  as the modulo divisor.

**Task**

Given a five digit integer, print the sum of its digits.

**Input Format**

The input contains a single five digit number, .

**Constraints**

**Output Format**

Print the sum of the digits of the five digit number.

**Sample Input 0**

10564

**Sample Output 0**

16

Program:-

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main() {

int num, sum = 0;

scanf("%d", &num);

while(num != 0) {

sum += num % 10;

num /= 10;

}

printf("%d", sum);

}

**Input (stdin):-**

**98999**

**Expected Output1:-**

**44**

**Input (stdin)**

**12345**

**Expected Output2:-**

**15**

**5C) /\*Given number is a prime or not. (Also Prime numbers between a given range)\*/**

#include <stdio.h>

#include <stdbool.h>

bool is\_prime(int number) {

int i;

if (number <= 1) {

return false;

}

if (number <= 3) {

return true;

}

if (number % 2 == 0 || number % 3 == 0) {

return false;

}

for (i = 5; i \* i <= number; i += 6) {

if (number % i == 0 || number % (i + 2) == 0) {

return false;

}

}

return true;

}

int main() {

int number,i;

printf("Enter a number: ");

scanf("%d", &number);

if (is\_prime(number)) {

printf("%d is a prime number.\n", number);

} else {

printf("%d is not a prime number.\n", number);

}

int start\_range, end\_range;

printf("Enter the start and end of the range: ");

scanf("%d %d", &start\_range, &end\_range);

printf("Prime numbers between %d and %d: ", start\_range, end\_range);

for (i = start\_range; i <= end\_range; i++) {

if (is\_prime(i)) {

printf("%d ", i);

}

}

printf("\n");

return 0;

}

**Expected Ouput1 :-**

Enter a number: 45

45 is not a prime number.

Enter the start and end of the range: 9

19

Prime numbers between 9 and 19: 11 13 17 19

**Expected Output2:-**

Enter a number: 11

11 is a prime number.

Enter the start and end of the range: 100

120

Prime numbers between 100 and 120: 101 103 107 109 113

**5D) /\*Armstrong Number or not.\*/**

#include <stdio.h>

#include <stdbool.h>

#include <math.h>

bool is\_armstrong(int number) {

int originalNumber, remainder, result = 0, n = 0;

originalNumber = number;

while (originalNumber != 0) {

originalNumber /= 10;

++n;

}

originalNumber = number;

while (originalNumber != 0) {

remainder = originalNumber % 10;

result += pow(remainder, n);

originalNumber /= 10;

}

return result == number;

}

int main() {

int number;

printf("Enter a number: ");

scanf("%d", &number);

if (is\_armstrong(number)) {

printf("%d is an Armstrong number.\n", number);

} else {

printf("%d is not an Armstrong number.\n", number);

}

return 0;

}

**ExxpectedOutput1:-**

Enter a number: 153

153 is an Armstrong number.

**Expected Output2:-**

Enter a number: 55

55 is not an Armstrong number.

**5E) Palindrome or not**

Program:-

#include <stdio.h>

#include <stdbool.h>

bool is\_palindrome(int number) {

int originalNumber, reversedNumber = 0, remainder;

originalNumber = number;

while (number > 0) {

remainder = number % 10;

reversedNumber = reversedNumber \* 10 + remainder;

number /= 10;

}

return originalNumber == reversedNumber;

}

int main() {

int number;

printf("Enter a number: ");

scanf("%d", &number);

if (is\_palindrome(number)) {

printf("%d is a palindrome number.\n", number);

} else {

printf("%d is not a palindrome number.\n", number);

}

return 0;

}

**Expected Output1:-**

Enter a number: 141

141 is a palindrome number.

**Expected Output2:-**

Enter a number: 213

213 is not a palindrome number.

5F)/\*Print a pattern of numbers from  to  as shown below. Each of the numbers is separated by a single space.\*/

4 4 4 4 4 4 4

4 3 3 3 3 3 4

4 3 2 2 2 3 4

4 3 2 1 2 3 4

4 3 2 2 2 3 4

4 3 3 3 3 3 4

4 4 4 4 4 4 4

**Input Format**

The input will contain a single integer .

**Constraints**

**Sample Input 0**

2

**Sample Output 0**

2 2 2

2 1 2

2 2 2

**Sample Input 1**

5

**Sample Output 1**

5 5 5 5 5 5 5 5 5

5 4 4 4 4 4 4 4 5

5 4 3 3 3 3 3 4 5

5 4 3 2 2 2 3 4 5

5 4 3 2 1 2 3 4 5

5 4 3 2 2 2 3 4 5

5 4 3 3 3 3 3 4 5

5 4 4 4 4 4 4 4 5

5 5 5 5 5 5 5 5 5

**Sample Input 2**

7

**Sample Output 2**

7 7 7 7 7 7 7 7 7 7 7 7 7

7 6 6 6 6 6 6 6 6 6 6 6 7

7 6 5 5 5 5 5 5 5 5 5 6 7

7 6 5 4 4 4 4 4 4 4 5 6 7

7 6 5 4 3 3 3 3 3 4 5 6 7

7 6 5 4 3 2 2 2 3 4 5 6 7

7 6 5 4 3 2 1 2 3 4 5 6 7

7 6 5 4 3 2 2 2 3 4 5 6 7

7 6 5 4 3 3 3 3 3 4 5 6 7

7 6 5 4 4 4 4 4 4 4 5 6 7

7 6 5 5 5 5 5 5 5 5 5 6 7

7 6 6 6 6 6 6 6 6 6 6 6 7

7 7 7 7 7 7 7 7 7 7 7 7 7

Program:-

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main() {

int n;

scanf("%d", &n);

int len = 2\*n - 1;

for (int i = 0; i < len; i++) {

for (int j = 0; j < len; j++) {

int min = i < j ? i : j;

min = min < len-i ? min : len-i-1;

min = min < len-j-1 ? min : len-j-1;

printf("%d ", n-min);

}

printf("\n");

}

return 0;

}

**Input (stdin)**

**2**

**Expected Output1:-**

**2 2 2**

**2 1 2**

**2 2 2**

**Input (stdin)**

**5**

**Expected Output2:-**

**5 5 5 5 5 5 5 5 5**

**5 4 4 4 4 4 4 4 5**

**5 4 3 3 3 3 3 4 5**

**5 4 3 2 2 2 3 4 5**

**5 4 3 2 1 2 3 4 5**

**5 4 3 2 2 2 3 4 5**

**5 4 3 3 3 3 3 4 5**

**5 4 4 4 4 4 4 4 5**

**5 5 5 5 5 5 5 5 5**

**5G)/\*\*/**

**Program:-**

#include <stdio.h>

int main() {

int rows, spaces, stars,i;

printf("Enter the number of rows: ");

scanf("%d", &rows);

for (i = 1; i <= rows; i++) {

// Print spaces

for (spaces = 1; spaces <= rows - i; spaces++) {

printf(" ");

}

// Print stars

for (stars = 1; stars <= 2 \* i - 1; stars++) {

printf("\*");

}

printf("\n");

}

return 0;

}

**Expected Output1:-**

Enter the number of rows: 5

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

**Expected Output2:-**

Enter the number of rows: 10

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Week 6A)

An array is a container object that holds a fixed number of values of a single type. To create an array in C, we can do int arr[n];. Here, arr, is a variable array which holds up to  integers. The above array is a static array that has memory allocated at compile time. A dynamic array can be created in C, using the malloc function and the memory is allocated on the heap at runtime. To create an integer array,  of size , int \*arr = (int\*)malloc(n \* sizeof(int)), where  points to the base address of the array. When you have finished with the array, use free(arr) to deallocate the memory.

In this challenge, create an array of size  dynamically, and read the values from stdin. Iterate the array calculating the sum of all elements. Print the sum and free the memory where the array is stored.

While it is true that you can sum the elements as they are read, without first storing them to an array, but you will not get the experience working with an array. Efficiency will be required later.

**Input Format**

The first line contains an integer, .  
The next line contains  space-separated integers.

**Constraints**

**Output Format**

Print the sum of the integers in the array.

**Sample Input 0**

6

16 13 7 2 1 12

**Sample Output 0**

51

**Sample Input 1**

7

1 13 15 20 12 13 2

**Sample Output 1**

76

Program:-

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int main() {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/

int n;

scanf("%d", &n);

int arr[n];

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

scanf("%d", &arr[i]);

}

int sum = 0;

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

sum += arr[i];

}

printf("%d", sum);

return 0;

}

**Input (stdin)**

**6**

**16 13 7 2 1 12**

**Expected Output1:-**

**51**

**Input (stdin)**

**8**

**15 5 16 15 17 11 5 11**

**Expected Output2:-**

**95**

**6B)** Given an array, of size , reverse it.

Example: If array, , after reversing it, the array should be, .

**Input Format**

The first line contains an integer, , denoting the size of the array. The next line contains  space-separated integers denoting the elements of the array.

**Constraints**

, where  is the  element of the array.

**Output Format**

The output is handled by the code given in the editor, which would print the array.

**Sample Input 0**

6

16 13 7 2 1 12

**Sample Output 0**

12 1 2 7 13 16

**Explanation 0**

Given array,  = . After reversing the array,  =

**Sample Input 1**

7

1 13 15 20 12 13 2

**Sample Output 1**

2 13 12 20 15 13 1

**Sample Input 2**

8

15 5 16 15 17 11 5 11

**Sample Output 2**

11 5 11 17 15 16 5 15

**Program:-**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int num, \*arr, i;

scanf("%d", &num);

arr = (int\*) malloc(num \* sizeof(int));

for(i = 0; i < num; i++) {

scanf("%d", arr + i);

}

/\* Write the logic to reverse the array. \*/

int temp;

for(i = 0; i < num/2; i++) {

temp = arr[i];

arr[i] = arr[num-1-i];

arr[num-1-i] = temp;

}

for(i = 0; i < num; i++)

printf("%d ", \*(arr + i));

return 0;

}

Input (stdin)

**6**

**16 13 7 2 1 12**

**Expected Output1:-**

**12 1 2 7 13 16**

Input (stdin)

**7**

**1 13 15 20 12 13 2**

**Expected Output2:-**

**2 13 12 20 15 13 1**

**Program:-**

#include <stdio.h>

int linearSearch(int arr[], int size, int target)

{

int i;

for (i = 0; i < size; i++) {

if (arr[i] == target) {

return i; // Return the index if the element is found

}

}

return -1; // Return -1 if the element is not found

}

int main() {

int arr[] = {12, 34, 56, 78, 90, 23, 45, 67};

int size = sizeof(arr) / sizeof(arr[0]);

int target;

printf("Enter the element to search: ");

scanf("%d", &target);

int result = linearSearch(arr, size, target);

if (result != -1) {

printf("Element found at index %d\n", result);

} else {

printf("Element not found in the array\n");

}

return 0;

}

**Expected Output 1:-**

Enter the element to search: 78

Element found at index 3

**Expected Output 2:-**

Enter the element to search: 98

Element not found in the array

**6D) /\*Find min and max elements in array\*/**

Program:-

#include <stdio.h>

int main() {

int arr[] = {12, 34, 56, 78, 90, 23, 45, 67};

int size = sizeof(arr) / sizeof(arr[0]);

int i;

int min = arr[0]; // Initialize min with the first element

int max = arr[0]; // Initialize max with the first element

for (i = 1; i < size; i++) {

if (arr[i] < min) {

min = arr[i]; // Update min if a smaller element is found

}

if (arr[i] > max) {

max = arr[i]; // Update max if a larger element is found

}

}

printf("Minimum element: %d\n", min);

printf("Maximum element: %d\n", max);

return 0;

}

**Expected Output 1:-**

Minimum element: 12

Maximum element: 90

**6E)/\* Insert an element into array\*/**

**Program:-**

#include <stdio.h>

int main() {

int arr[] = {12, 34, 56, 78, 90, 23, 45, 67};

int size = sizeof(arr) / sizeof(arr[0]);

int i;

int element\_to\_insert;

int position;

printf("Enter the element to insert: ");

scanf("%d", &element\_to\_insert);

printf("Enter the position to insert (0 to %d): ", size);

scanf("%d", &position);

if (position < 0 || position > size) {

printf("Invalid position. Please enter a position between 0 and %d.\n", size);

return 1; // Exit with an error code

}

int new\_size = size + 1;

int new\_arr[new\_size];

// Copy elements before the insertion point

for (i = 0; i < position; i++) {

new\_arr[i] = arr[i];

}

// Insert the new element

new\_arr[position] = element\_to\_insert;

// Copy elements after the insertion point

for (i = position; i < size; i++) {

new\_arr[i + 1] = arr[i];

}

printf("Array after insertion:\n");

for (i = 0; i < new\_size; i++) {

printf("%d ", new\_arr[i]);

}

printf("\n");

return 0;

}

**Expected Output 1:-**

Enter the element to insert: 88

Enter the position to insert (0 to 8): 8

Array after insertion:

12 34 56 78 90 23 45 67 88

**Expected Output2**:-

Enter the element to insert: 90

Enter the position to insert (0 to 8): 0

Array after insertion:

90 12 34 56 78 90 23 45 67

6F) /\*Eliminate duplicate elements from array\*/

Program:-

#include <stdio.h>

int main() {

int arr[] = {12, 34, 56, 78, 34, 23, 45, 78};

int size = sizeof(arr) / sizeof(arr[0]);

int i,j;

// Initialize a new array to store unique elements

int unique\_arr[size];

int unique\_size = 0;

for (i = 0; i < size; i++) {

int isDuplicate = 0;

// Check if the current element is a duplicate

for (j = 0; j < unique\_size; j++) {

if (arr[i] == unique\_arr[j]) {

isDuplicate = 1;

break;

}

}

// If it's not a duplicate, add it to the unique array

if (!isDuplicate) {

unique\_arr[unique\_size] = arr[i];

unique\_size++;

}

}

printf("Original array: ");

for (i = 0; i < size; i++) {

printf("%d ", arr[i]);

}

printf("\n");

printf("Array after eliminating duplicates: ");

for (i = 0; i < unique\_size; i++) {

printf("%d ", unique\_arr[i]);

}

printf("\n");

return 0;

}

**Excepted Output:-**

**Original array: 12 34 56 78 34 23 45 78**

**Array after eliminating duplicates: 12 34 56 78 23 45**

**6G) /\*Sorting of elements in an array using Bubble sort\*/**

**Program:-**

#include <stdio.h>

void bubbleSort(int arr[], int size) {

int temp,i,j;

for (i = 0; i < size - 1; i++) {

for (j = 0; j < size - 1 - i; j++) {

if (arr[j] > arr[j + 1]) {

// Swap arr[j] and arr[j + 1]

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

int main() {

int arr[] = {64, 34, 25, 12, 22, 11, 90};

int size = sizeof(arr) / sizeof(arr[0]);

int i;

printf("Original array: ");

for (i = 0; i < size; i++) {

printf("%d ", arr[i]);

}

printf("\n");

bubbleSort(arr, size);

printf("Array after Bubble Sort: ");

for (i = 0; i < size; i++) {

printf("%d ", arr[i]);

}

printf("\n");

return 0;

}

**Expected Output:-**

Original array: 64 34 25 12 22 11 90

Array after Bubble Sort: 11 12 22 25 34 64 90

Week7:

7A) Sum of two 2-D arrays

Program:-

#include <stdio.h>

int main() {

int rows, cols;

int i,j;

// Input the dimensions of the arrays

printf("Enter the number of rows: ");

scanf("%d", &rows);

printf("Enter the number of columns: ");

scanf("%d", &cols);

int arr1[rows][cols], arr2[rows][cols], sum[rows][cols];

// Input the elements of the first array

printf("Enter elements of the first array:\n");

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

scanf("%d", &arr1[i][j]);

}

}

// Input the elements of the second array

printf("Enter elements of the second array:\n");

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

scanf("%d", &arr2[i][j]);

}

}

// Calculate the sum of the two arrays

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

sum[i][j] = arr1[i][j] + arr2[i][j];

}

}

// Display the sum of the arrays

printf("Sum of the two arrays:\n");

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

printf("%d ", sum[i][j]);

}

printf("\n");

}

return 0;

}

Expected Output1:-

Enter the number of rows: 2

Enter the number of columns: 3

Enter elements of the first array:

2

3

3

4

5

5

Enter elements of the second array:

5

3

2

3

4

5

Sum of the two arrays:

7 6 5

7 9 10

**ExpectedOutput 2:-**

Enter the number of rows: 2

Enter the number of columns: 2

Enter elements of the first array:

3

4

5

45

Enter elements of the second array:

3

2

2

2

Sum of the two arrays:

6 6

7 47

**7B)/\*** Multiplication of two 2-D arrays**\*/**

**Program:-**

#include <stdio.h>

int main() {

int rows1, cols1, rows2, cols2;

int i,j,k;

// Input the dimensions of the first matrix

printf("Enter the number of rows of the first matrix: ");

scanf("%d", &rows1);

printf("Enter the number of columns of the first matrix: ");

scanf("%d", &cols1);

// Input the dimensions of the second matrix

printf("Enter the number of rows of the second matrix: ");

scanf("%d", &rows2);

printf("Enter the number of columns of the second matrix: ");

scanf("%d", &cols2);

if (cols1 != rows2) {

printf("Matrix multiplication is not possible. Columns of the first matrix must be equal to rows of the second matrix.\n");

return 1; // Exit with an error code

}

int matrix1[rows1][cols1], matrix2[rows2][cols2], result[rows1][cols2];

// Input the elements of the first matrix

printf("Enter elements of the first matrix:\n");

for (i = 0; i < rows1; i++) {

for (j = 0; j < cols1; j++) {

scanf("%d", &matrix1[i][j]);

}

}

// Input the elements of the second matrix

printf("Enter elements of the second matrix:\n");

for (i = 0; i < rows2; i++) {

for (j = 0; j < cols2; j++) {

scanf("%d", &matrix2[i][j]);

}

}

// Initialize the result matrix to zeros

for (i = 0; i < rows1; i++) {

for (j = 0; j < cols2; j++) {

result[i][j] = 0;

}

}

// Perform matrix multiplication

for (i = 0; i < rows1; i++) {

for (j = 0; j < cols2; j++) {

for (k = 0; k < cols1; k++) {

result[i][j] += matrix1[i][k] \* matrix2[k][j];

}

}

}

// Display the result matrix

printf("Result of matrix multiplication:\n");

for (i = 0; i < rows1; i++) {

for (j = 0; j < cols2; j++) {

printf("%d ", result[i][j]);

}

printf("\n");

}

return 0;

}

**Expected output1:-**

Enter the number of rows of the first matrix: 2

Enter the number of columns of the first matrix: 2

Enter the number of rows of the second matrix: 2

Enter the number of columns of the second matrix: 2

Enter elements of the first matrix:

2

3

5

5

Enter elements of the second matrix:

4

3

3

3

Result of matrix multiplication:

17 15

35 30

**Expected Output2:-**

Enter the number of rows of the first matrix: 3

Enter the number of columns of the first matrix: 3

Enter the number of rows of the second matrix: 3

Enter the number of columns of the second matrix: 3

Enter elements of the first matrix:

5

45

2

1

2

3

5

6

55

Enter elements of the second matrix:

5

5

5

5

7

7

7

7

5

Result of matrix multiplication:

264 354 350

36 40 34

440 452 342

**7C)/\*** Transpose of a Matrix**\*/**

**Program:-**

#include <stdio.h>

int main() {

int rows, cols;

int i,j;

// Input the dimensions of the matrix

printf("Enter the number of rows: ");

scanf("%d", &rows);

printf("Enter the number of columns: ");

scanf("%d", &cols);

int matrix[rows][cols], transpose[cols][rows];

// Input the elements of the matrix

printf("Enter elements of the matrix:\n");

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

scanf("%d", &matrix[i][j]);

}

}

// Calculate the transpose

for (i = 0; i < cols; i++) {

for (j = 0; j < rows; j++) {

transpose[i][j] = matrix[j][i];

}

}

// Display the original matrix

printf("Original matrix:\n");

for (i = 0; i < rows; i++) {

for (j = 0; j < cols; j++) {

printf("%d ", matrix[i][j]);

}

printf("\n");

}

// Display the transpose

printf("Transpose of the matrix:\n");

for (i = 0; i < cols; i++) {

for (j = 0; j < rows; j++) {

printf("%d ", transpose[i][j]);

}

printf("\n");

}

return 0;

}

**Expected Output1:-**

**Enter the number of rows: 2**

**Enter the number of columns: 2**

**Enter elements of the matrix:**

**3**

**56**

**5**

**8**

**Original matrix:**

**3 56**

**5 8**

**Transpose of the matrix:**

**3 5**

**56 8**

**Expected Output2:-**

**Enter the number of rows: 3**

**Enter the number of columns: 3**

**Enter elements of the matrix:**

**1**

**2**

**3**

**45**

**6**

**8**

**8**

**5**

**5**

**Original matrix:**

**1 2 3**

**45 6 8**

**8 5 5**

**Transpose of the matrix:**

**1 45 8**

**2 6 5**

**3 8 5**

**7D)/\*** Trace of a Matrix**\*/**

**Program:-**

#include <stdio.h>

int main() {

int m, n,i,j;

// Input the dimensions of the matrix

printf("Enter the number of rows: ");

scanf("%d", &m);

printf("Enter the number of columns: ");

scanf("%d", &n);

if (m != n) {

printf("The trace is only defined for square matrices (m x m).\n");

return 1; // Exit with an error code

}

int matrix[m][n];

int trace = 0;

// Input the elements of the matrix

printf("Enter elements of the matrix:\n");

for (i = 0; i < m; i++) {

for (j = 0; j < n; j++) {

scanf("%d", &matrix[i][j]);

if (i == j) {

trace += matrix[i][j];

}

}

}

// Display the matrix

printf("Matrix:\n");

for (i = 0; i < m; i++) {

for (j = 0; j < n; j++) {

printf("%d ", matrix[i][j]);

}

printf("\n");

}

// Display the trace

printf("Trace of the matrix: %d\n", trace);

return 0;

}

**Expected Output1:-**

Enter the number of rows: 2

Enter the number of columns: 2

Enter elements of the matrix:

6

5

2

5

Matrix:

6 5

2 5

Trace of the matrix: 11

**Expected Output2:-**

Enter the number of rows: 4

Enter the number of columns: 9

The trace is only defined for square matrices (m x m).

7E) /\*Lower Triangular Matrix\*/

Program:-

#include <stdio.h>

int main() {

int n,i,j;

// Input the size of the square matrix

printf("Enter the size of the square matrix: ");

scanf("%d", &n);

int matrix[n][n];

// Input the elements of the matrix

printf("Enter elements of the square matrix:\n");

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

scanf("%d", &matrix[i][j]);

}

}

// Check if the matrix is lower triangular

int isLowerTriangular = 1;

for (i = 0; i < n; i++) {

for (j = i + 1; j < n; j++) {

if (matrix[i][j] != 0) {

isLowerTriangular = 0;

break;

}

}

}

// Display the matrix

printf("Square matrix:\n");

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

printf("%d ", matrix[i][j]);

}

printf("\n");

}

// Display the result

if (isLowerTriangular) {

printf("The matrix is lower triangular.\n");

} else {

printf("The matrix is not lower triangular.\n");

}

return 0;

}

Expected Output1:-

Enter the size of the square matrix: 3

Enter elements of the square matrix:

1

0

0

4

2

0

6

5

1

Square matrix:

1 0 0

4 2 0

6 5 1

The matrix is lower triangular.

**Expected Output 2:-**

Enter the size of the square matrix: 2

Enter elements of the square matrix:

1

2

2

3

Square matrix:

1 2

2 3

The matrix is not lower triangular

7.a) Sum of two 2-D arrays

#include <stdio.h>

int main() {

int row, col;

printf("Enter the number of rows and columns for the 2-D arrays: ");

scanf("%d %d", &row, &col);

// Declare two 2-D arrays with the same dimensions

int array1[row][col], array2[row][col], sum[row][col];

// Input elements for the first array

printf("Enter elements for the first array:\n");

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

scanf("%d", &array1[i][j]);

}

}

// Input elements for the second array

printf("Enter elements for the second array:\n");

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

scanf("%d", &array2[i][j]);

}

}

// Calculate the sum of the two arrays

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

sum[i][j] = array1[i][j] + array2[i][j];

}

}

// Display the sum array

printf("Sum of the two arrays:\n");

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

printf("%d\t", sum[i][j]);

}

printf("\n");

}

return 0;

}

Output

Enter the number of rows and columns for the 2-D arrays: 2 2

Enter elements for the first array:

1 2

3 4

Enter elements for the second array:

5 6

7 8

Sum of the two arrays:

6 8

10 12

7.b)

#include <stdio.h>

int main() {

int row1, col1, row2, col2;

printf("Enter the number of rows and columns for the first matrix: ");

scanf("%d %d", &row1, &col1);

printf("Enter the number of rows and columns for the second matrix: ");

scanf("%d %d", &row2, &col2);

if (col1 != row2) {

printf("Matrix multiplication is not possible. The number of columns in the first matrix must be equal to the number of rows in the second matrix.\n");

return 1;

}

// Declare two 2-D arrays

int matrix1[row1][col1], matrix2[row2][col2], result[row1][col2];

// Input elements for the first matrix

printf("Enter elements for the first matrix:\n");

for (int i = 0; i < row1; i++) {

for (int j = 0; j < col1; j++) {

scanf("%d", &matrix1[i][j]);

}

}

// Input elements for the second matrix

printf("Enter elements for the second matrix:\n");

for (int i = 0; i < row2; i++) {

for (int j = 0; j < col2; j++) {

scanf("%d", &matrix2[i][j]);

}

}

// Initialize the result matrix with zeros

for (int i = 0; i < row1; i++) {

for (int j = 0; j < col2; j++) {

result[i][j] = 0;

}

}

// Perform matrix multiplication

for (int i = 0; i < row1; i++) {

for (int j = 0; j < col2; j++) {

for (int k = 0; k < col1; k++) {

result[i][j] += matrix1[i][k] \* matrix2[k][j];

}

}

}

// Display the result matrix

printf("Result of matrix multiplication:\n");

for (int i = 0; i < row1; i++) {

for (int j = 0; j < col2; j++) {

printf("%d\t", result[i][j]);

}

printf("\n");

}

return 0;

}

Output

Enter the number of rows and columns for the first matrix: 2 2

Enter elements for the first matrix:

2 3

4 5

Enter the number of rows and columns for the second matrix: 2 2

Enter elements for the second matrix:

6 7

8 9

Result of matrix multiplication:

26 30

50 58

7.c) Transpose of a Matrix

#include <stdio.h>

int main() {

int row, col;

printf("Enter the number of rows and columns for the matrix: ");

scanf("%d %d", &row, &col);

// Declare a 2-D array for the matrix

int matrix[row][col], transpose[col][row];

// Input elements for the matrix

printf("Enter elements for the matrix:\n");

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

scanf("%d", &matrix[i][j]);

}

}

// Calculate the transpose of the matrix

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

transpose[j][i] = matrix[i][j];

}

}

// Display the transpose matrix

printf("Transpose of the matrix:\n");

for (int i = 0; i < col; i++) {

for (int j = 0; j < row; j++) {

printf("%d\t", transpose[i][j]);

}

printf("\n");

}

return 0;

}

Output

Enter the size of the square matrix: 3

Enter elements for the square matrix:

1 2 3

4 5 6

7 8 9

Trace of the matrix: 15

7.d) Trace of a Matrix

#include <stdio.h>

int main() {

int row, col;

printf("Enter the number of rows and columns for the matrix: ");

scanf("%d %d", &row, &col);

// Declare a 2-D array for the matrix

int matrix[row][col], transpose[col][row];

// Input elements for the matrix

printf("Enter elements for the matrix:\n");

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

scanf("%d", &matrix[i][j]);

}

}

// Calculate the transpose of the matrix

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

transpose[j][i] = matrix[i][j];

}

}

// Display the transpose matrix

printf("Transpose of the matrix:\n");

for (int i = 0; i < col; i++) {

for (int j = 0; j < row; j++) {

printf("%d\t", transpose[i][j]);

}

printf("\n");

}

return 0;

}

Output

Enter the size of the square matrix: 3

Enter elements for the square matrix:

1 2 3

4 5 6

7 8 9

Trace of the matrix: 15

7.e) Lower Triangular Matrix

#include <stdio.h>

int main() {

int size;

printf("Enter the size of the square matrix: ");

scanf("%d", &size);

// Declare a 2-D array for the lower triangular matrix

int matrix[size][size];

// Initialize the lower triangular matrix

for (int i = 0; i < size; i++) {

for (int j = 0; j < size; j++) {

if (i >= j) {

matrix[i][j] = 1; // Set the elements below or on the main diagonal to 1

} else {

matrix[i][j] = 0; // Set the elements above the main diagonal to 0

}

}

}

// Display the lower triangular matrix

printf("Lower Triangular Matrix:\n");

for (int i = 0; i < size; i++) {

for (int j = 0; j < size; j++) {

printf("%d\t", matrix[i][j]);

}

printf("\n");

}

return 0;

}

8.b) Count number of alphabets (lowercase, uppercase, consonants, vowels) and digits

#include <stdio.h>

#include <ctype.h>

int main() {

char input[100];

int lowercase = 0, uppercase = 0, vowels = 0, consonants = 0, digits = 0;

printf("Enter a string: ");

gets(input); // You can use gets or fgets to read the input string.

for (int i = 0; input[i] != '\0'; i++) {

if (isalpha(input[i])) {

if (islower(input[i])) {

lowercase++;

if (input[i] == 'a' || input[i] == 'e' || input[i] == 'i' || input[i] == 'o' || input[i] == 'u' ||

input[i] == 'A' || input[i] == 'E' || input[i] == 'I' || input[i] == 'O' || input[i] == 'U') {

vowels++;

} else {

consonants++;

}

} else if (isupper(input[i])) {

uppercase++;

if (input[i] == 'A' || input[i] == 'E' || input[i] == 'I' || input[i] == 'O' || input[i] == 'U') {

vowels++;

} else {

consonants++;

}

}

} else if (isdigit(input[i])) {

digits++;

}

}

printf("Lowercase letters: %d\n", lowercase);

printf("Uppercase letters: %d\n", uppercase);

printf("Vowels: %d\n", vowels);

printf("Consonants: %d\n", consonants);

printf("Digits: %d\n", digits);

return 0;

}

Output

Enter a string: Hello, World! 123

Lowercase letters: 8

Uppercase letters: 2

Vowels: 3

Consonants: 7

Digits: 3

8.c) Lowercase to Uppercase, Uppercase to Lowercase, Toggle case, Sentential case

#include <stdio.h>

#include <string.h>

void toUpperCase(char \*str) {

for (int i = 0; str[i]; i++) {

if (str[i] >= 'a' && str[i] <= 'z') {

str[i] = str[i] - 32; // Convert to uppercase

}

}

}

void toLowerCase(char \*str) {

for (int i = 0; str[i]; i++) {

if (str[i] >= 'A' && str[i] <= 'Z') {

str[i] = str[i] + 32; // Convert to lowercase

}

}

}

void toggleCase(char \*str) {

for (int i = 0; str[i]; i++) {

if (str[i] >= 'a' && str[i] <= 'z') {

str[i] = str[i] - 32; // Convert to uppercase

} else if (str[i] >= 'A' && str[i] <= 'Z') {

str[i] = str[i] + 32; // Convert to lowercase

}

}

}

void sententialCase(char \*str) {

int capitalize = 1; // Start with a capital letter

for (int i = 0; str[i]; i++) {

if (str[i] >= 'a' && str[i] <= 'z' && capitalize) {

str[i] = str[i] - 32; // Convert to uppercase

capitalize = 0;

} else if (str[i] >= 'A' && str[i] <= 'Z' && !capitalize) {

str[i] = str[i] + 32; // Convert to lowercase

capitalize = 1;

}

}

}

int main() {

char input[100];

printf("Enter a string: ");

fgets(input, sizeof(input), stdin);

input[strcspn(input, "\n")] = '\0'; // Remove the newline character from input

char output[100];

strcpy(output, input); // Copy the input to output for each case conversion

toUpperCase(output);

printf("Uppercase: %s\n", output);

strcpy(output, input);

toLowerCase(output);

printf("Lowercase: %s\n", output);

strcpy(output, input);

toggleCase(output);

printf("Toggle Case: %s\n", output);

strcpy(output, input);

sententialCase(output);

printf("Sentential Case: %s\n", output);

return 0;

}

Output

Enter a string: This is a Sample String

Uppercase: THIS IS A SAMPLE STRING

Lowercase: this is a sample string

Toggle Case: tHIS IS A sAMPLE sTRING

Sentential Case: This is A sample String

8.e) Find string length, concatenate 2 strings, reverse a string using built-in and without built-in string functions.

#include <stdio.h>

#include <string.h>

// Function to find the length of a string without using built-in function

int customStringLength(const char \*str) {

int length = 0;

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

length++;

}

return length;

}

// Function to concatenate two strings without using built-in function

void customStringConcat(char \*destination, const char \*source) {

int destLength = customStringLength(destination);

int sourceLength = customStringLength(source);

for (int i = 0; i < sourceLength; i++) {

destination[destLength + i] = source[i];

}

destination[destLength + sourceLength] = '\0';

}

// Function to reverse a string without using built-in function

void customStringReverse(char \*str) {

int length = customStringLength(str);

for (int i = 0; i < length / 2; i++) {

char temp = str[i];

str[i] = str[length - i - 1];

str[length - i - 1] = temp;

}

}

int main() {

char str1[100], str2[100], concatenated[200];

printf("Enter a string: ");

scanf("%s", str1);

printf("Enter another string: ");

scanf("%s", str2);

// Finding the length of str1

int length1 = strlen(str1);

printf("Using built-in function: Length of str1 = %d\n", length1);

// Concatenating str1 and str2 using built-in function

strcpy(concatenated, str1);

strcat(concatenated, str2);

printf("Using built-in function: Concatenated string = %s\n", concatenated);

// Reversing str1 using built-in function

strcpy(concatenated, str1);

strrev(concatenated); // Note: strrev is not a standard C function, but some compilers provide it.

printf("Using built-in function: Reversed str1 = %s\n", concatenated);

// Finding the length of str1 without using built-in function

length1 = customStringLength(str1);

printf("Without built-in function: Length of str1 = %d\n", length1);

// Concatenating str1 and str2 without using built-in function

strcpy(concatenated, str1);

customStringConcat(concatenated, str2);

printf("Without built-in function: Concatenated string = %s\n", concatenated);

// Reversing str1 without using built-in function

strcpy(concatenated, str1);

customStringReverse(concatenated);

printf("Without built-in function: Reversed str1 = %s\n", concatenated);

return 0;

}

Output:

Enter a string: Hello

Enter another string: World

Using built-in function: Length of str1 = 5

Using built-in function: Concatenated string = HelloWorld

Using built-in function: Reversed str1 = olleH

Without built-in function: Length of str1 = 5

Without built-in function: Concatenated string = HelloWorld

Without built-in function: Reversed str1 = olleH

10.d) Find the sum of a 1D array using malloc()

#include <stdio.h>

#include <stdlib.h>

int main() {

int n;

printf("Enter the number of elements in the array: ");

scanf("%d", &n);

// Dynamically allocate memory for the array

int \*arr = (int \*)malloc(n \* sizeof(int));

if (arr == NULL) {

printf("Memory allocation failed. Exiting...\n");

return 1; // Exit the program with an error code

}

// Input elements into the array

printf("Enter the elements of the array:\n");

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

scanf("%d", &arr[i]);

}

// Calculate the sum of elements

int sum = 0;

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

sum += arr[i];

}

printf("Sum of the elements in the array: %d\n", sum);

// Free the dynamically allocated memory

free(arr);

return 0;

}

Output

Enter the number of elements in the array: 5

Enter the elements of the array:

1

2

3

4

5

Sum of the elements in the array: 15

10.e) Swap two numbers using functions and pointers - call by value and reference.

#include <stdio.h>

// Function to swap two numbers using call by value

void swapByValue(int a, int b) {

int temp = a;

a = b;

b = temp;

}

// Function to swap two numbers using call by reference

void swapByReference(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main() {

int num1, num2;

printf("Enter two numbers: ");

scanf("%d %d", &num1, &num2);

// Swap by value

printf("Before swapping by value: num1 = %d, num2 = %d\n", num1, num2);

swapByValue(num1, num2);

printf("After swapping by value: num1 = %d, num2 = %d\n", num1, num2);

// Swap by reference using pointers

printf("Before swapping by reference: num1 = %d, num2 = %d\n", num1, num2);

swapByReference(&num1, &num2);

printf("After swapping by reference: num1 = %d, num2 = %d\n", num1, num2);

return 0;

}

11.a) Write a C program to find the total, average of n students using structures

#include <stdio.h>

// Define a structure for student information

struct Student {

char name[50];

int rollNumber;

float marks;

};

int main() {

int n;

printf("Enter the number of students: ");

scanf("%d", &n);

// Declare an array of structures to store student information

struct Student students[n];

// Input student information

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

printf("Enter details for student %d:\n", i + 1);

printf("Name: ");

scanf("%s", students[i].name);

printf("Roll Number: ");

scanf("%d", &students[i].rollNumber);

printf("Marks: ");

scanf("%f", &students[i].marks);

}

// Calculate the total marks and average marks

float totalMarks = 0.0;

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

totalMarks += students[i].marks;

}

float averageMarks = totalMarks / n;

printf("Total marks of all students: %.2f\n", totalMarks);

printf("Average marks of all students: %.2f\n", averageMarks);

return 0;

}

Output

Enter the number of students: 2

Enter details for student 1:

Name: John

Roll Number: 101

Marks: 85.5

Enter details for student 2:

Name: Jane

Roll Number: 102

Marks: 92.0

Total marks of all students: 177.50

Average marks of all students: 88.75

11.d) Copy one structure variable to another structure of the same type.

#include <stdio.h>

// Define a structure

struct Student {

char name[50];

int rollNumber;

float marks;

};

int main() {

// Declare two structure variables of the same type

struct Student student1, student2;

// Initialize student1

printf("Enter details for student 1:\n");

printf("Name: ");

scanf("%s", student1.name);

printf("Roll Number: ");

scanf("%d", &student1.rollNumber);

printf("Marks: ");

scanf("%f", &student1.marks);

// Copy student1 to student2

student2 = student1;

// Display the contents of student2

printf("\nStudent 2 (copied from Student 1):\n");

printf("Name: %s\n", student2.name);

printf("Roll Number: %d\n", student2.rollNumber);

printf("Marks: %.2f\n", student2.marks);

return 0;

}

Output

Enter details for student 1:

Name: John

Roll Number: 101

Marks: 85.5

Student 2 (copied from Student 1):

Name: John

Roll Number: 101

Marks: 85.50

11.e) Read student name and marks from the command line and display the student details along with the total.

#include <stdio.h>

#include <stdlib.h>

int main(int argc, char \*argv[]) {

if (argc != 3) {

printf("Usage: %s <Student Name> <Marks>\n", argv[0]);

return 1;

}

// Read student name and marks from command line arguments

char \*name = argv[1];

float marks = atof(argv[2]);

// Display student details

printf("Student Name: %s\n", name);

printf("Marks: %.2f\n", marks);

return 0;

}

Output:

$ ./student\_details John 85.5

Student Name: John

Marks: 85.50

11.f) Shift/rotate using bitfields.

#include <stdio.h>

// Define a bitfield structure

struct Bitfield {

unsigned int data : 8; // 8-bit field for demonstration

};

int main() {

struct Bitfield bf;

bf.data = 0x0A; // Initialize with binary 00001010

printf("Original Value: 0x%X\n", bf.data);

// Right-shift operation

bf.data = bf.data >> 1;

printf("Right-Shift by 1: 0x%X\n", bf.data);

// Left-shift operation

bf.data = bf.data << 2;

printf("Left-Shift by 2: 0x%X\n", bf.data);

// Bitwise rotation to the left

bf.data = (bf.data << 3) | (bf.data >> (8 - 3));

printf("Left-Rotate by 3: 0x%X\n", bf.data);

// Bitwise rotation to the right

bf.data = (bf.data >> 2) | (bf.data << (8 - 2));

printf("Right-Rotate by 2: 0x%X\n", bf.data);

return 0;

}

Output:

Original Value: 0xA

Right-Shift by 1: 0x5

Left-Shift by 2: 0x14

Left-Rotate by 3: 0x41

Right-Rotate by 2: 0x8

12.a) Write text into and read text from a file

#include <stdio.h>

#include <string.h>

int main() {

FILE \*file;

char fileName[] = "sample.txt"; // File name

// Write text into the file

file = fopen(fileName, "w");

if (file == NULL) {

printf("Unable to open the file for writing.\n");

return 1;

}

char textToWrite[] = "This is a sample text that we're writing to a file.";

fputs(textToWrite, file);

fclose(file);

printf("Text written to the file.\n");

// Read text from the file

file = fopen(fileName, "r");

if (file == NULL) {

printf("Unable to open the file for reading.\n");

return 1;

}

char textFromFile[1000]; // Assuming a maximum of 1000 characters

fgets(textFromFile, sizeof(textFromFile), file);

fclose(file);

printf("Text read from the file:\n%s\n", textFromFile);

return 0;

}

Output

Text written to the file.

Text read from the file:

This is a sample text that we're writing to a file.

12.c) Copy the contents of one file to another file.

#include <stdio.h>

#include <stdlib.h>

int main() {

FILE \*sourceFile, \*destinationFile;

char sourceFileName[] = "source.txt"; // Change to your source file name

char destinationFileName[] = "destination.txt"; // Change to your destination file name

char ch;

// Open the source file for reading

sourceFile = fopen(sourceFileName, "r");

if (sourceFile == NULL) {

printf("Unable to open the source file.\n");

return 1;

}

// Open the destination file for writing

destinationFile = fopen(destinationFileName, "w");

if (destinationFile == NULL) {

printf("Unable to open the destination file.\n");

fclose(sourceFile);

return 1;

}

// Copy the contents from source to destination

while ((ch = fgetc(sourceFile)) != EOF) {

fputc(ch, destinationFile);

}

printf("File contents copied successfully.\n");

// Close the files

fclose(sourceFile);

fclose(destinationFile);

return 0;

}

Output

File contents copied successfully.

12.d) Merge two files into the third file using command-line arguments.

#include <stdio.h>

#include <stdlib.h>

int main(int argc, char \*argv[]) {

if (argc != 4) {

printf("Usage: %s <InputFile1> <InputFile2> <OutputFile>\n", argv[0]);

return 1;

}

FILE \*file1, \*file2, \*outputFile;

char \*inputFileName1 = argv[1];

char \*inputFileName2 = argv[2];

char \*outputFileName = argv[3];

char ch;

// Open the first input file for reading

file1 = fopen(inputFileName1, "r");

if (file1 == NULL) {

printf("Unable to open the first input file.\n");

return 1;

}

// Open the second input file for reading

file2 = fopen(inputFileName2, "r");

if (file2 == NULL) {

printf("Unable to open the second input file.\n");

fclose(file1);

return 1;

}

// Open the output file for writing

outputFile = fopen(outputFileName, "w");

if (outputFile == NULL) {

printf("Unable to open the output file.\n");

fclose(file1);

fclose(file2);

return 1;

}

// Copy the contents from the first input file to the output file

while ((ch = fgetc(file1)) != EOF) {

fputc(ch, outputFile);

}

// Copy the contents from the second input file to the output file

while ((ch = fgetc(file2)) != EOF) {

fputc(ch, outputFile);

}

printf("Files merged successfully.\n");

// Close the files

fclose(file1);

fclose(file2);

fclose(outputFile);

return 0;

}

Output

./merge\_files inputfile1.txt inputfile2.txt outputfile.txt

12.e) Find no. of lines, words and characters in a file

#include <stdio.h>

#include <stdlib.h>

int main() {

FILE \*file;

char fileName[] = "sample.txt"; // Replace with the file name you want to analyze

char ch;

int lines = 0, words = 0, characters = 0;

int inWord = 0; // Flag to track whether currently in a word

// Open the file for reading

file = fopen(fileName, "r");

if (file == NULL) {

printf("Unable to open the file for reading.\n");

return 1;

}

while ((ch = fgetc(file)) != EOF) {

characters++;

// Count words and lines

if (ch == '\n') {

lines++;

}

if (ch == ' ' || ch == '\n' || ch == '\t') {

inWord = 0;

} else if (inWord == 0) {

inWord = 1;

words++;

}

}

// Close the file

fclose(file);

// Display the counts

printf("Number of lines: %d\n", lines);

printf("Number of words: %d\n", words);

printf("Number of characters: %d\n", characters);

return 0;

}

Output:

Number of lines: 5

Number of words: 22

Number of characters: 139