**NAME: Ranjith M (BGSW/PJ-ETA-FTP-C)**

**NTID: MFJ1COB**

**LAB ASSIGNMENT – 2**

**QUESTION 1:**

Write a C program to accept two inputs from user Temperature in Celsius & Category (Human / Dog / Cat). Output as Fahrenheit and Kelvin & output the category is in fever or not.

**AIM:**

To write a C program for converting Celsius to Fahrenheit and output whether they have fever for specific category.

**PSUEDOCODE:**

Define constants CATEGORY\_H, CATEGORY\_D, CATEGORY\_C, CATEGORY\_h, CATEGORY\_d, CATEGORY\_c

Function celsiusToFahrenheit(celsius):

Return (9.0F / 5.0F) \* celsius + 32.0F

Function celsiusToKelvin(celsius):

Return celsius + 273.15F

Function isFever(temperature, category):

If category is CATEGORY\_H or CATEGORY\_h:

If temperature is between 98.6F and 99.0F:

Return 1

Else if category is CATEGORY\_D or CATEGORY\_d:

If temperature is between 101.0F and 102.5F:

Return 1

Else if category is CATEGORY\_C or CATEGORY\_c:

If temperature is between 99.0F and 102.5F:

Return 1

Return 0

Function main():

Declare celsius, fahrenheit, kelvin as floats

Declare category as char

Declare input as character array

Print "Enter the temperature in Celsius:"

Read celsius from the user using fgets and sscanf

Print "Enter the category (H for Human, D for Dog, C for Cat):"

Read category from the user using fgets and sscanf

fahrenheit = celsiusToFahrenheit(celsius)

kelvin = celsiusToKelvin(celsius)

int fever = isFever(fahrenheit, category)

Print "------------------------------------------------------"

Print "Equivalent Fahrenheit: " + fahrenheit

Print "Equivalent Kelvin: " + kelvin

If fever is 1:

Print "The category is in fever range."

Else:

Print "The category is not in fever range."

Print "------------------------------------------------------"

Return 0

**ALGORITHM:**

**Step 1:** Define constants for the different categories:

* CATEGORY\_H and CATEGORY\_h for Human.
* CATEGORY\_D and CATEGORY\_d for Dog.
* CATEGORY\_C and CATEGORY\_c for Cat.

**Step 2:** Create functions for temperature conversion:

* celsiusToFahrenheit(celsius) to convert Celsius to Fahrenheit using the formula: (9.0F / 5.0F) \* celsius + 32.0F.
* celsiusToKelvin(celsius) to convert Celsius to Kelvin using the formula: celsius + 273.15F.

**Step 3:** Create a function to check if the temperature is within the fever range based on the category:

* isFever(temperature, category)
* If the category is Human (H or h), check if the temperature is between 98.6°F and 99.0°F.
* If the category is Dog (D or d), check if the temperature is between 101.0°F and 102.5°F.
* If the category is Cat (C or c), check if the temperature is between 99.0°F and 102.5°F.
* If the category is not valid, return 0 (false).

**Step 4:** In the main() function,

* Declare variables celsius, fahrenheit, and kelvin as floats to store temperature values, and category as a char to store the input category.
* Create a character array input to store user input.
* Print "Enter the temperature in Celsius:" to prompt the user for input.
* Read the temperature in Celsius from the user using fgets() and sscanf().
* Print "Enter the category (H for Human, D for Dog, C for Cat):" to prompt the user for input.
* Read the category from the user using fgets() .
* Convert Celsius to Fahrenheit and Kelvin using the corresponding functions.
* Check if the category is in the fever range using the isFever() function.
* Print the converted Fahrenheit and Kelvin values, along with whether the category is in the fever range or not.

**Step 5:** Return 0 to indicate successful execution of the main() function.

**PROGRAM:**

#include <stdio.h>

#define CATEGORY\_H 'H'

#define CATEGORY\_D 'D'

#define CATEGORY\_C 'C'

#define CATEGORY\_h 'h'

#define CATEGORY\_d 'd'

#define CATEGORY\_c 'c'

// Function to convert Celsius to Fahrenheit

float celsiusToFahrenheit(float celsius) {

    return (9.0F / 5.0F) \* celsius + 32.0F;

}

// Function to convert Celsius to Kelvin

float celsiusToKelvin(float celsius) {

    return celsius + 273.15F;

}

// Function to check if the temperature is in fever range based on category

int isFever(float temperature, char category) {

    if (category == CATEGORY\_H || category == CATEGORY\_h) { // Human

        return (temperature >= 98.6F && temperature <= 99.0F);

    } else if (category == CATEGORY\_D || category == CATEGORY\_d) { // Dog

        return (temperature >= 101.0F && temperature <= 102.5F);

    } else if (category == CATEGORY\_C || category == CATEGORY\_c) { // Cat

        return (temperature >= 99.0F && temperature <= 102.5F);

    } else {

        return 0; // Invalid category

    }

}

int main(void) {

    float celsius, fahrenheit, kelvin;

    char input[100]; // Buffer for user input

    char category;

    // Input from the user for temperature in Celsius

    printf("Enter the temperature in Celsius: ");

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

    sscanf(input, "%f", &celsius);

    // Input from the user for category

    printf("Enter the category (H for Human, D for Dog, C for Cat): ");

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

    // Convert Celsius to Fahrenheit and Kelvin

    fahrenheit = celsiusToFahrenheit(celsius);

    kelvin = celsiusToKelvin(celsius);

    // Check if the category is in fever range

    int fever = isFever(fahrenheit, category);

    printf("%d",fever);

    // Output the results

    printf("------------------------------------------------------\n");

    printf("Equivalent Fahrenheit: %.2f\n", fahrenheit);

    printf("Equivalent Kelvin: %.2f\n", kelvin);

    if (fever) {

        printf("The category is in fever range.\n");

    } else {

        printf("The category is not in fever range.\n");

    }

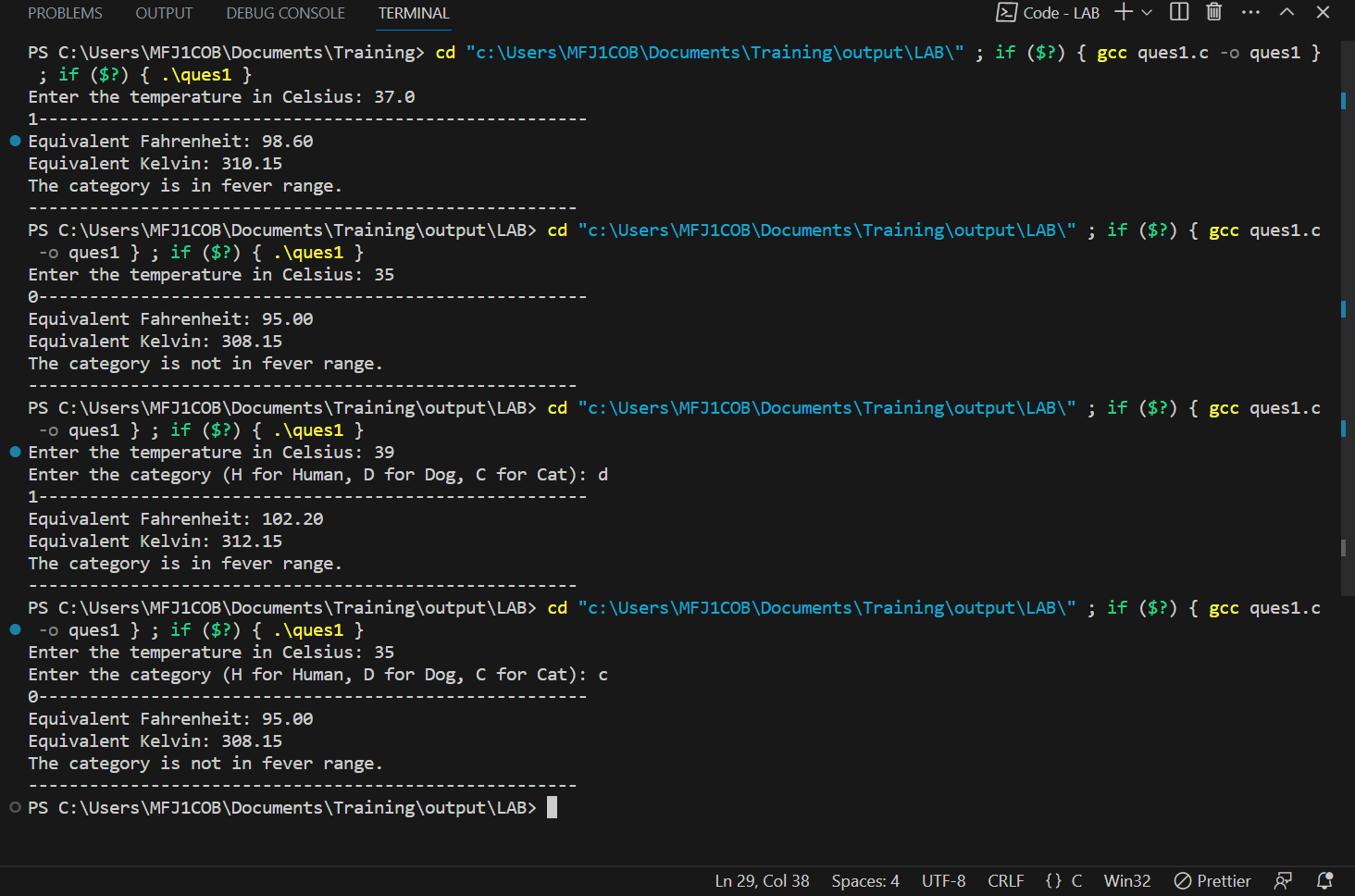
    printf("------------------------------------------------------\n");

    return 0;

}

**OUTPUT:**

Output is shown for different scenarios.

****

**QUESTION 2:**

Write a C program to count Odd and Even numbers and include a function to check whether it has at least one even or odd numbers.

**AIM:**

**To accept ten number as input and count number of odd and even numbers in the input also include a function to check whether it contains at least one even or odd number.**

**PSUEDOCODE:**

Constant ARRAY\_SIZE = 10

Function hasBothEvenAndOdd(arr, size):

Declare hasEven and hasOdd as integers and set them to 0

for i from 0 to size-1:

If arr[i] is even:

Set hasEven to 1

Else:

Set hasOdd to 1

Return hasEven AND hasOdd

Function main():

Declare numbers as an integer array of size ARRAY\_SIZE

Declare evenCount and oddCount as integers and set them to 0

Print "Enter ten numbers:"

Read ten numbers from the user and store them in the numbers array using a loop

for i from 0 to ARRAY\_SIZE-1:

If numbers[i] is even:

Increment evenCount

Else:

Increment oddCount

If hasBothEvenAndOdd(numbers, ARRAY\_SIZE) is true:

Print "---------------------------------------------------"

Print "Number of even numbers: " + evenCount

Print "Number of odd numbers: " + oddCount

Print "---------------------------------------------------"

Else:

Print "---------------------------------------------------"

Print "Invalid Inputs: All numbers are either even or odd."

Print "---------------------------------------------------"

Return 0

**ALGORITHM:**

**Step 1:** Define a constant ARRAY\_SIZE with a value of 10.

**Step 2:** Create a function hasBothEvenAndOdd(arr, size) to check if the given array has at least one even and one odd number:

* Initialize two flags, hasEven and hasOdd, to 0.
* Loop through the array from index 0 to index size-1.
* For each element, if it is even (i.e., arr[i] % 2 == 0), set hasEven to 1.
* If the element is odd, set hasOdd to 1.
* After the loop, return the logical AND of hasEven and hasOdd.

**Step 3:** In the main() function,

* Declare an integer array numbers with size ARRAY\_SIZE.
* Declare evenCount and oddCount as integers to count the number of even and odd numbers, respectively.
* Print "Enter ten numbers:" to prompt the user for input.
* Read ten numbers from the user and store them in the numbers array using a loop.
* Count the number of even and odd numbers by looping through the numbers array and incrementing the respective counters.
* Call the hasBothEvenAndOdd(numbers, ARRAY\_SIZE) function to check if the array has both even and odd numbers also for 0 and 1.
* If the function returns true (i.e., there is at least one even and one odd number), print the counts of even and odd numbers.
* If the function returns false (i.e., all numbers are either even or odd), print a message indicating invalid inputs.

**Step 4:** Return 0 to indicate successful execution of the main() function.

**PROGRAM:**

#include <stdio.h>

#define ARRAY\_SIZE 10

// Function to check if at least one even and one odd number is present in the array

int hasBothEvenAndOdd(const int arr[], int size) {

    int hasEven = 0;

    int hasOdd = 0;

    for (int i = 0; i < size; i++) {

        if (arr[i] == 0) {

            hasEven = 1;

        } else if (arr[i] == 1) {

            hasOdd = 1;

        } else if (arr[i] % 2 == 0) {

            hasEven = 1;

        } else {

            hasOdd = 1;

        }

        // Break the loop early if both even and odd numbers are found

        if (hasEven && hasOdd) {

            break;

        }

    }

    return hasEven && hasOdd;

}

int main(void) {

    int numbers[ARRAY\_SIZE];

    int evenCount = 0;

    int oddCount = 0;

    printf("Enter ten numbers:\n");

    for (int i = 0; i < ARRAY\_SIZE; i++) {

        scanf("%d", &numbers[i]);

    }

    // Count the number of even and odd numbers

    for (int i = 0; i < ARRAY\_SIZE; i++) {

        if (numbers[i] % 2 == 0) {

            evenCount++;

        } else {

            oddCount++;

        }

    }

    // Check if there is at least one even and one odd number

    if (hasBothEvenAndOdd(numbers, ARRAY\_SIZE)) {

        printf("---------------------------------------------------\n");

        printf("Number of even numbers: %d\n", evenCount);

        printf("Number of odd numbers: %d\n", oddCount);

        printf("---------------------------------------------------\n");

    } else {

        printf("---------------------------------------------------\n");

        printf("Invalid Inputs: All numbers are either even or odd.\n");

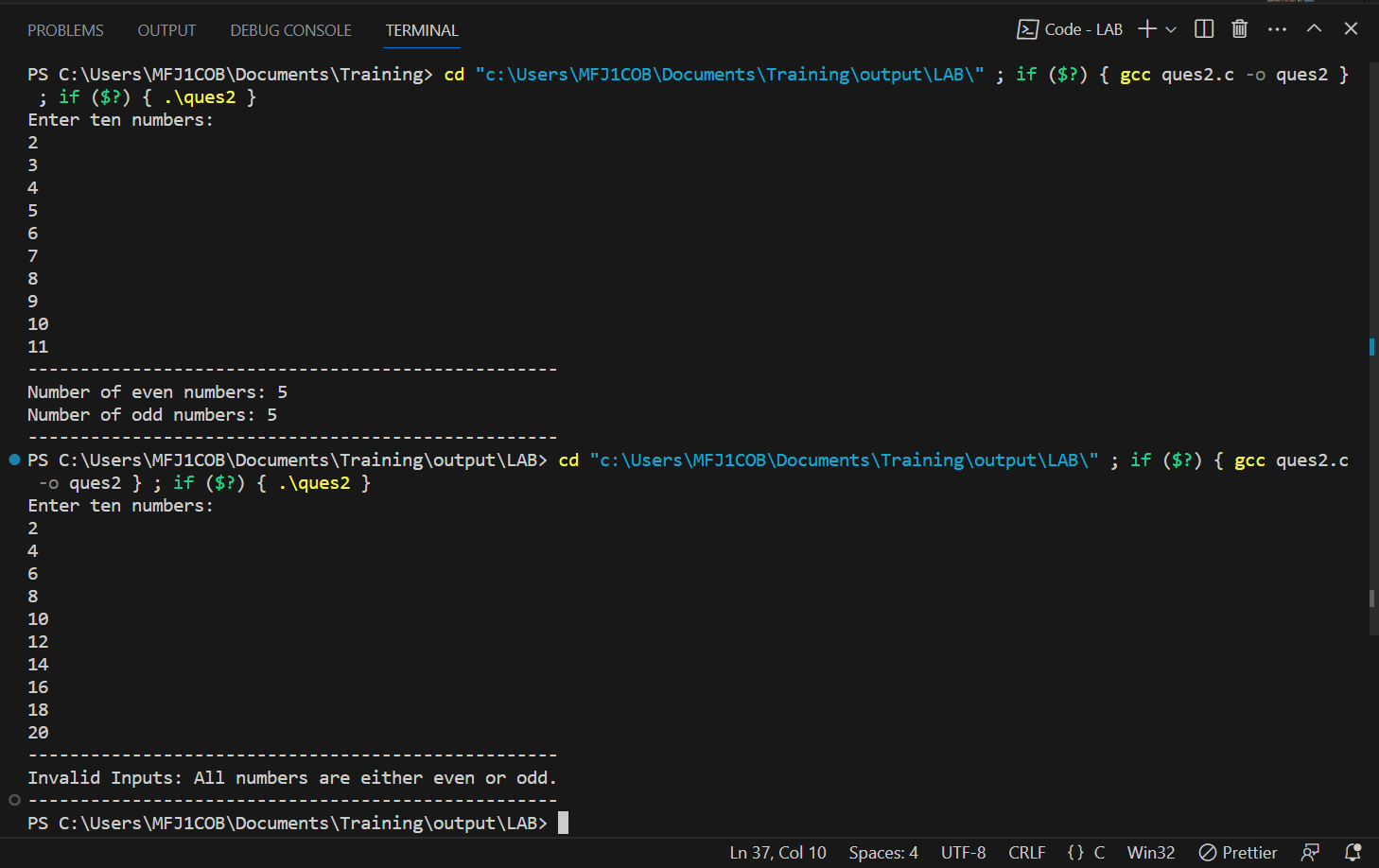
        printf("---------------------------------------------------\n");

    }

    return 0;

}

**OUTPUT:**



**QUESTION 3:**

To write a C program that prints the results for addition, subtraction, multiplication, and remainder.

**AIM:**

To write a C program that performs all the arithmetic operations and print the results.

**ALGORITHM:**

**Step 1:** Create a function called performOperations(num1, num2, add, multiply, subtract, remainder) to perform arithmetic operations on the given numbers:

* Declare add, multiply, subtract, and remainder as integer pointers.
* Calculate the addition of num1 and num2 and store the result in the memory location pointed by add.
* Calculate the multiplication of num1 and num2 and store the result in the memory location pointed by multiply.
* Calculate the subtraction of num2 from num1 and store the result in the memory location pointed by subtract.
* Calculate the remainder of num1 divided by num2 and store the result in the memory location pointed by remainder.

**Step 2:** In the main() function,

* Declare num1, num2, addition, multiplication, subtraction, and remainder as integers.
* Print "Enter the first positive number:" to prompt the user for input.
* Read num1 from the user using scanf().
* Print "Enter the second positive number:" to prompt the user for input.
* Read num2 from the user using scanf().
* Check if both num1 and num2 are positive (i.e., greater than 0).
* If any of the numbers is not positive, print "Invalid inputs: Both numbers should be positive." and return 1 to indicate an error.
* Call the performOperations() function to calculate the results and store them in the respective variables using pointers.
* Print the results of addition, multiplication, subtraction, and remainder.

**Step 3:** Return 0 to indicate successful execution of the main() function.

**PSUEDOCODE:**

Function performOperations(num1, num2, add, multiply, subtract, remainder):

Declare add, multiply, subtract, and remainder as integer pointers

Calculate num1 + num2 and store the result in the memory location pointed by add

Calculate num1 \* num2 and store the result in the memory location pointed by multiply

Calculate num1 - num2 and store the result in the memory location pointed by subtract

Calculate num1 % num2 and store the result in the memory location pointed by remainder

Function main():

Declare num1, num2, addition, multiplication, subtraction, and remainder as integers

Print "Enter the first positive number:"

Read num1 from the user using scanf

Print "Enter the second positive number:"

Read num2 from the user using scanf

If num1 <= 0 OR num2 <= 0:

Print "Invalid inputs: Both numbers should be positive."

Return 1

PerformOperations(num1, num2, &addition, &multiplication, &subtraction, &remainder)

Print "Addition: " + addition

Print "Multiplication: " + multiplication

Print "Subtraction: " + subtraction

Print "Remainder: " + remainder

Return 0

**PROGRAM:**

#include <stdio.h>

// Function to perform addition, multiplication, subtraction, and remainder

void performOperations(const int num1, const int num2, int \*add, int \*multiply, int \*subtract, int \*remainder) {

    \*add = num1 + num2;

    \*multiply = num1 \* num2;

    \*subtract = num1 - num2;

    \*remainder = num1 % num2;

}

int main(void) {

    int num1, num2;

    int addition = 0, multiplication = 0, subtraction = 0, remainder = 0;

    // Input from the user

    printf("Enter the first positive number: ");

    scanf("%d", &num1);

    printf("Enter the second positive number: ");

    scanf("%d", &num2);

    // Check if inputs are positive

    if (num1 <= 0 || num2 <= 0) {

        printf("Invalid inputs: Both numbers should be positive.\n");

        return 1;

    }

    // Perform operations using pointers

    performOperations(num1, num2, &addition, &multiplication, &subtraction, &remainder);

    // Output the results

    printf("Addition: %d\n", addition);

    printf("Multiplication: %d\n", multiplication);

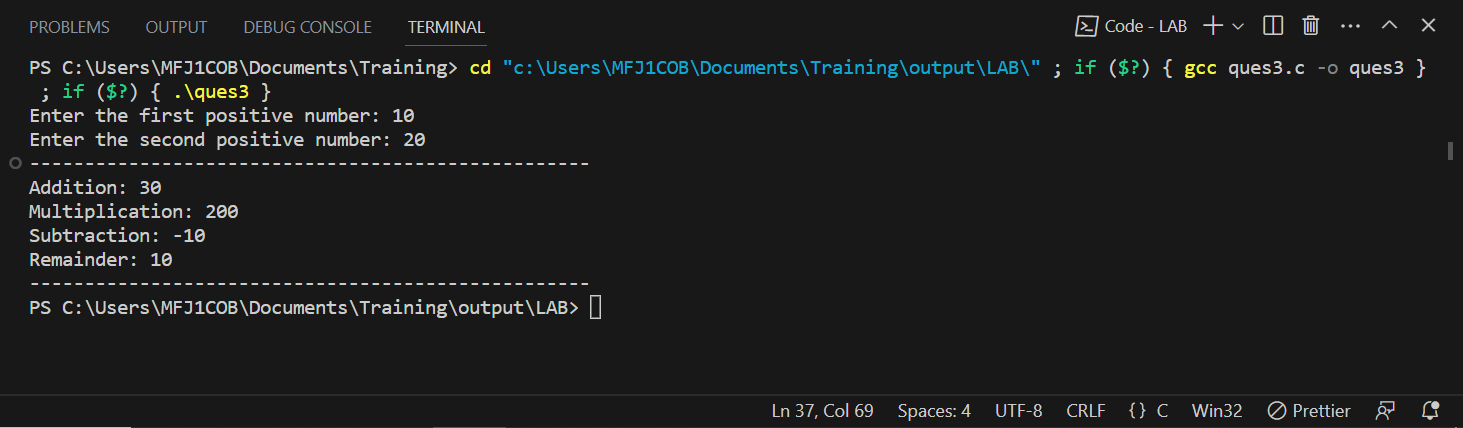
    printf("Subtraction: %d\n", subtraction);

    printf("Remainder: %d\n", remainder);

    return 0;

}

**OUTPUT:**

****

**QUESTION 4:**

Write a C program to take the word as input string and generates the output as print one character only one time.

**AIM:**

To write a C program that print the results by eliminating the occurrence more than one.

**PSUEDOCODE:**

Constant MAX\_LENGTH = 100

Function removeDuplicates(input):

Calculate the length of the input string and store it in len

If len is 0:

Print "Input string is empty."

Return

Declare output as a character array of size MAX\_LENGTH and initialize it with null characters

Declare outputIndex as an integer and set it to 0

Loop i from 0 to len-1:

Loop j from 0 to outputIndex-1:

If input[i] is equal to output[j]:

Break // Character already in the output string

If j is equal to outputIndex:

If outputIndex < MAX\_LENGTH - 1:

Add input[i] to output at the position outputIndex

Increment outputIndex

Else:

Break // Output buffer is full

Null-terminate the output array at the position outputIndex

Print "---------------------------------------------------"

Print "Output: " + output

Print "---------------------------------------------------"

Function main():

Declare input as a character array of size MAX\_LENGTH

Print "Enter the word: "

Read input from the user using fgets and store it in the input array

Remove the newline character (if any) from input by replacing it with the null character '\0'

Call removeDuplicates(input) to remove duplicate characters and print the resulting word

Return 0

**ALGORITHM:**

**Step 1:** Define a constant MAX\_LENGTH with a value of 100 to specify the maximum length of the input and output strings.

**Step 2:** Create a function called removeDuplicates(input) to remove duplicate characters from the given input string:

* Calculate the length of the input string using strlen().
* If the length of the input string is 0, print "Input string is empty." and return.
* Create an output character array output of size MAX\_LENGTH and initialize it with null characters.
* Initialize an integer outputIndex to 0, which will be used to keep track of the next available index in the output array.
* Loop through each character in the input string.
* For each character, check if it is already present in the output array.
* If the character is not present in the output array, add it to the output array at the outputIndex position and increment outputIndex.
* If the output array is full (i.e., outputIndex is equal to MAX\_LENGTH - 1), break out of the loop.
* Null-terminate the output array at the position outputIndex to mark the end of the resulting string.
* Print the resulting output string.

**Step 3:** In the main() function,

* Declare an array input of characters with size MAX\_LENGTH.
* Print "Enter the word: " to prompt the user for input.
* Read the input word from the user using fgets() and store it in the input array.
* Remove the newline character (if any) from the input word by replacing it with the null character '\0'.
* Call the removeDuplicates() function with the input word as an argument to remove duplicate characters and print the resulting word.

**Step 4:** Return 0 to indicate successful execution of the main() function.

**PROGRAM:**

#include <stdio.h>

#include <string.h>

#define MAX\_LENGTH 100

// Function to remove duplicate characters and print the result

void removeDuplicates(const char \*input)

{

    int len = strlen(input);

    if (len == 0)

    {

        printf("Input string is empty.\n");

        return;

    }

    char output[MAX\_LENGTH] = {0};

    int outputIndex = 0;

    for (int i = 0; i < len; i++)

    {

        int j;

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

            if (input[i] == output[j])

            {

                break; // Character already in the output string

            }

        }

        if (j == outputIndex)

        {

            if (outputIndex < MAX\_LENGTH - 1)

            {

                output[outputIndex++] = input[i]; // Add character to the output

            }

            else

            {

                // Output buffer is full, break out of the loop

                break;

            }

        }

    }

    output[outputIndex] = '\0'; // Null-terminate the output string

    printf("---------------------------------------------------\n");

    printf("Output: %s\n", output);

    printf("---------------------------------------------------\n");

}

int main(void)

{

    char input[MAX\_LENGTH];

    printf("Enter the word: ");

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

    // Remove the newline character from input

    int len = strlen(input);

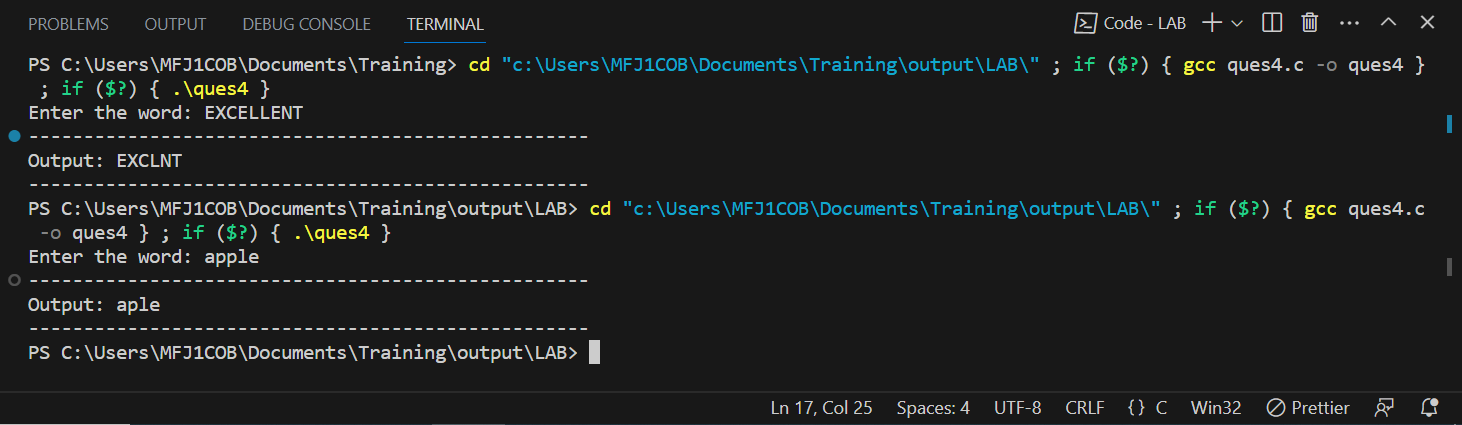
    input[strcspn(input, "\n")] = '\0';

    removeDuplicates(input);

    return 0;

}

**OUTPUT:**

****