**Reg. No: 21BCE1297 Name: Vidhi Shah**

**PPS1**

**Q1**

**Aim:**

Write a program in C to find the roots of a quadratic equation.

**Procedure:**

**Input:**

Coefficients a, b and c

**Output:**

Roots

**Algorithm:**

Step 1: Read a, b and c

Step 2: Discriminant, d = b^2 - 4ac

Step 3: If discriminant is greater than 0

Then roots are real and distinct

Root1 = (-b + sqrt(d))/2a, Root2 = (-b - sqrt(d))/2a

Step 4: If discriminant is equal to 0

Then roots are real and equal

Root1 = Root2 = -b/2a

Step 3: If discriminant is less than 0

Then roots are imaginary

Real Part = -b/2a, Imaginary Part = sqrt(-d)/2a

Root1 = realPart + imagPart(i), Root2 = realPart – imagPart(i)

Step 6: Display the roots

**Code:**

#include <stdio.h>

#include <math.h>

int main() {

    double a, b, c, d, r1, r2, realPart, imagPart;

    printf("\nCoefficients a, b and c are: ");

    scanf("%lf %lf %lf", &a, &b, &c);

    d = pow(b,2) - 4 \* a \* c;

    // Real and distinct roots

    if (d > 0) {

        r1 = (-b + sqrt(d)) / (2 \* a);

        r2 = (-b - sqrt(d)) / (2 \* a);

        printf("\nQuadratic Equation has real and distinct roots:\n");

        printf("Root 1 = %.2lf and Root 2 = %.2lf\n\n", r1, r2);

    }

    // Real and equal roots

    else if (d == 0) {

        r1 = r2 = -b / (2 \* a);

        printf("\nQuadratic Equation has real and equal roots:\n");

        printf("Root 1 = Root 2 = %.2lf\n\n", r1);

    }

    // Imaginary roots

    else {

        realPart = -b / (2 \* a);

        imagPart = sqrt(-d) / (2 \* a);

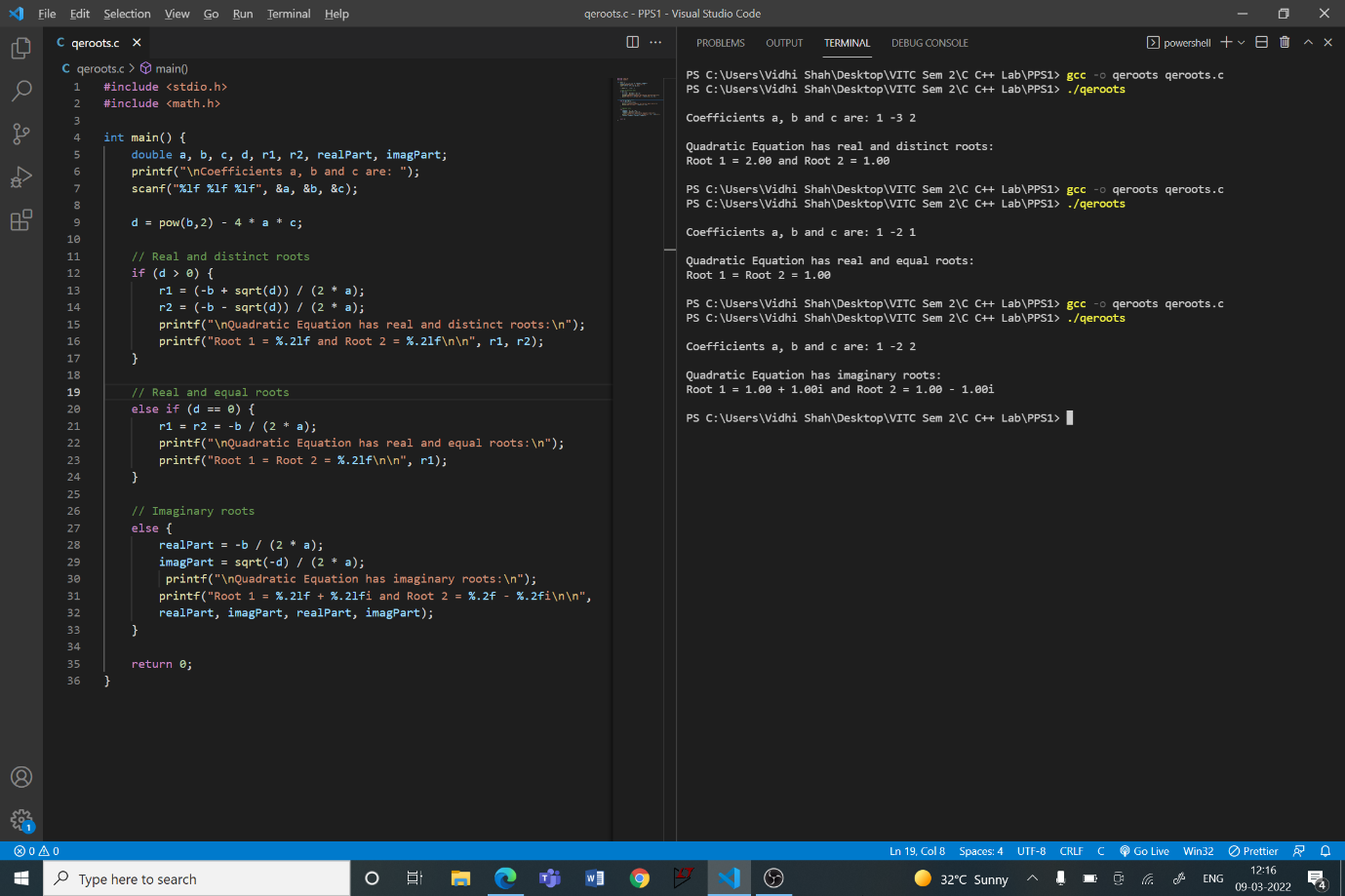
         printf("\nQuadratic Equation has imaginary roots:\n");

        printf("Root 1 = %.2lf + %.2lfi and Root 2 = %.2f - %.2fi\n\n",

        realPart, imagPart, realPart, imagPart);

    }

    return 0;

****}

**Q2**

**Aim:**

Write a program in C to create a simple calculator.

**Procedure:**

**Input:**

Operator, op

Operands, n1 and n2

**Output:**

Equation with answer

**Algorithm:**

Step 1: Read operator and operands, op, n1, n2

Step 2: Use switch case for operator

Case 1 (‘+’): Result = n1 + n2

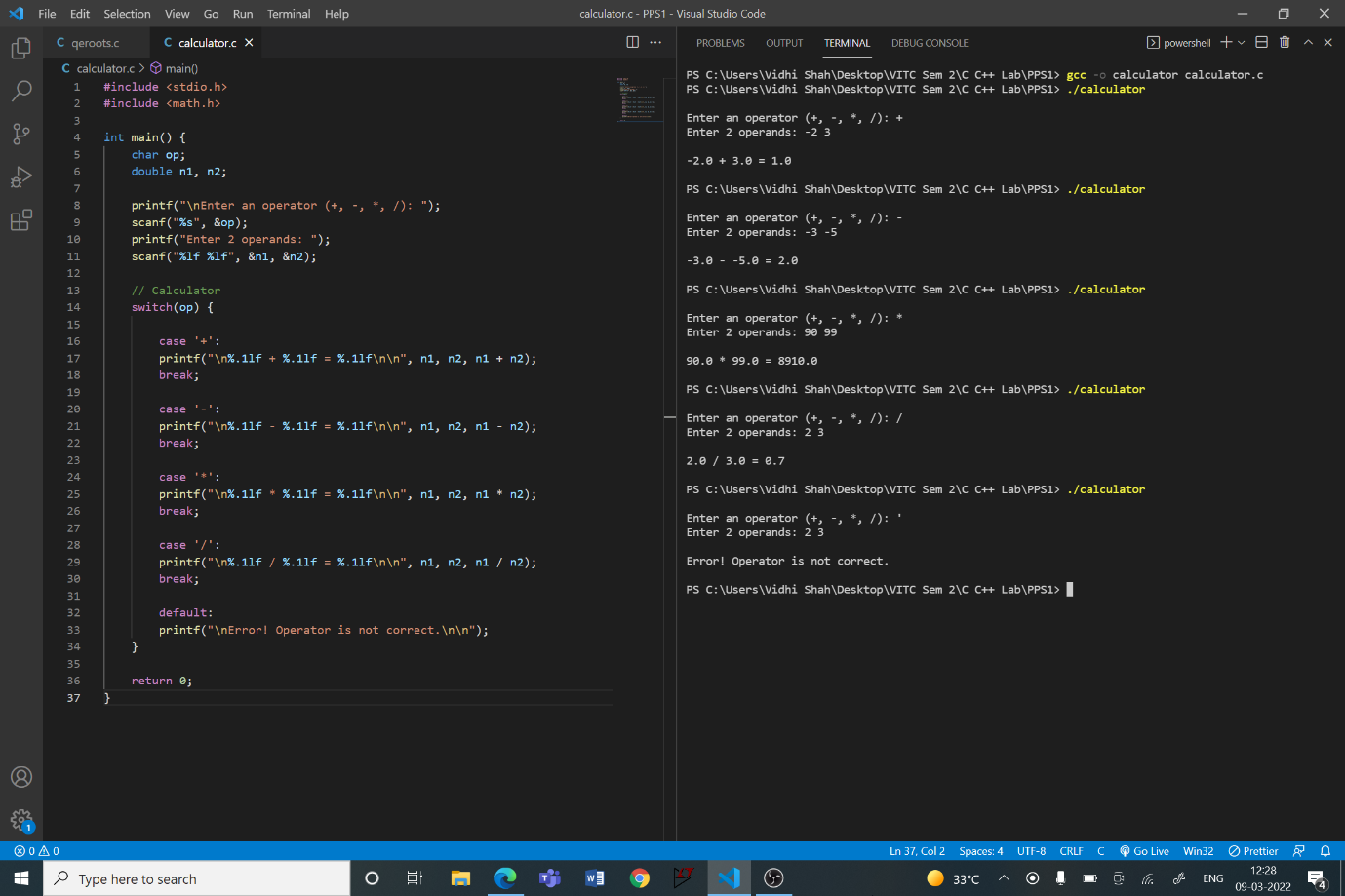
Case 2 (‘-’): Result = n1 - n2

Case 3 (‘\*’): Result = n1 \* n2

Case 4 (‘/’): Result = n1 / n2

Default: Error message for invalid input

Step 3: Print the result

**Code:**

**Q3**

**Aim:**

Write a program in C to display the days of the week getting the user input as a character indicating the first letter of the day. Demonstrate the use of Switch case statements.

**Procedure:**

**Input:**

Character indicating the first letter of the day, wd

**Output:**

Day

**Algorithm:**

Step 1: Read character, wd

Step 2: Use switch case for character

Case 1 (‘M’, ‘m’): Day = Monday

Case 2 (‘T’): Day = Tuesday

Case 3 (‘W’, ‘w’): Day = Wednesday

Case 4 (‘t’): Day = Thursday

Case 5 (‘F’, ‘f’): Day = Friday

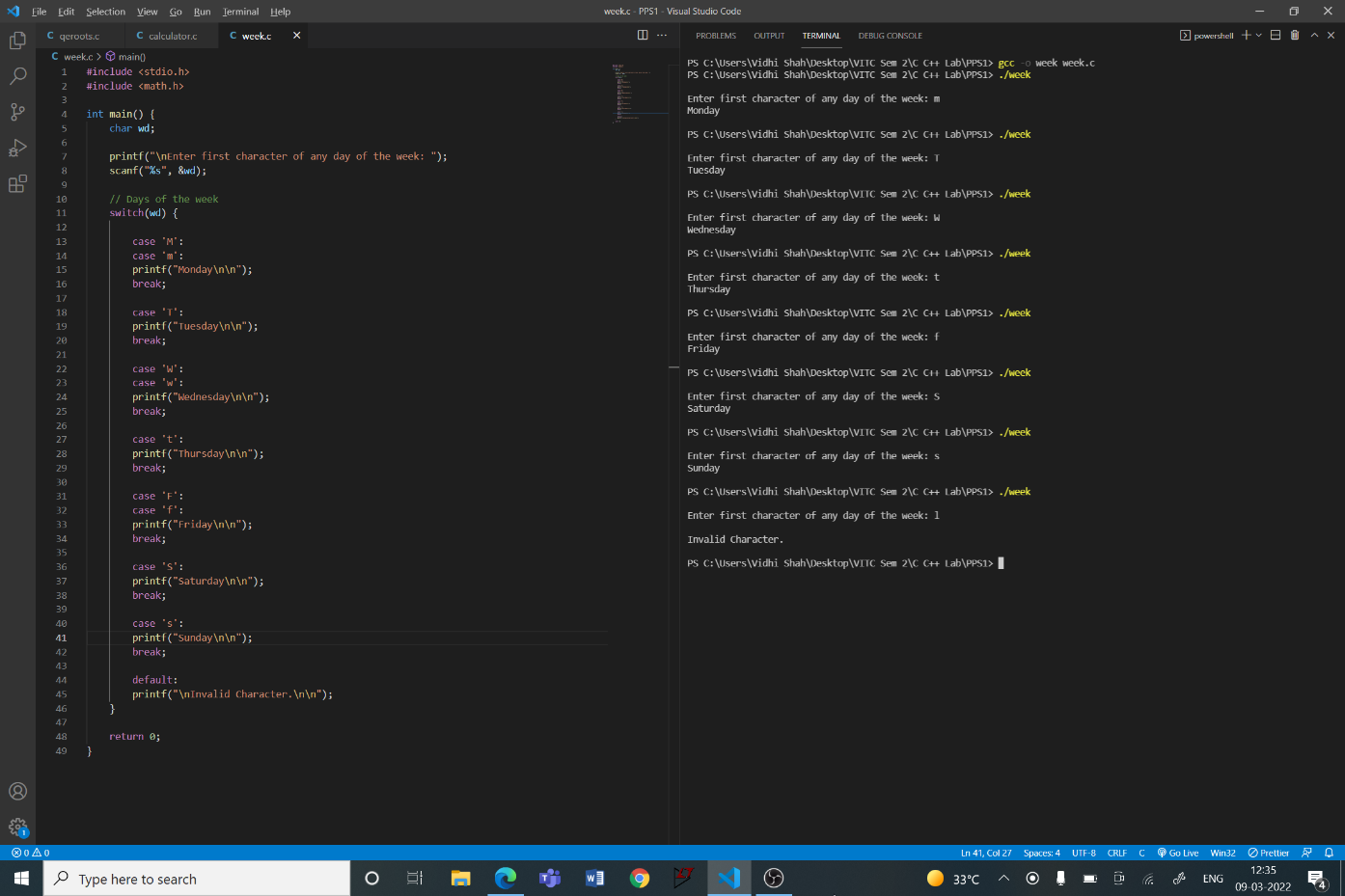
Case 6 (‘S’): Day = Saturday

Case 7 (‘s’): Day = Sunday

Default: Error message for invalid input

Step 6: Print the day

**Code:**

****

**PPS2**

**Q1**

**Aim:**

Write a program in C using while loop structure to display the sum of first n natural numbers.

**Procedure:**

**Input:**

Natural number, n

**Output:**

Sum of n natural numbers

**Algorithm:**

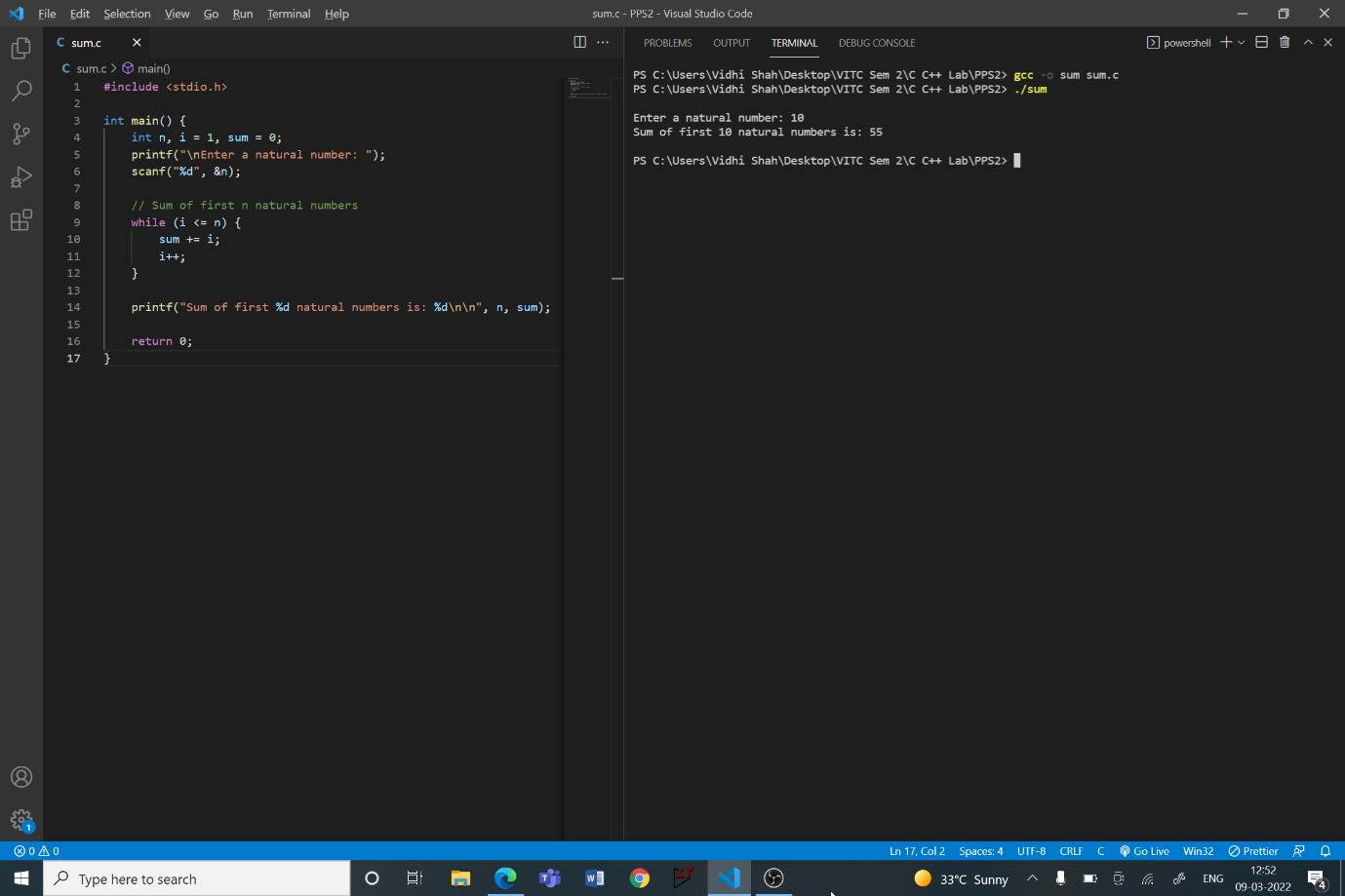
Step 1: Read n, initialize i to 1 and sum to 0

Step 2: Using while loop until i is less than equal to n

Add i to sum

Increment i by 1

Step 3: Print the sum

**Code:**

**Q2**

**Aim:**

Write a program in C using for loop structure to find the sum and average of n numbers.

**Procedure:**

**Input:**

Number of elements, n

Next n lines contain n numbers

**Output:**

Sum of the numbers

Average of the numbers

**Algorithm:**

Step 1: Read n, initialise sum to 0

Step 2: Using for loop initialise i to one. Until i is less than or equal to n

Read a number

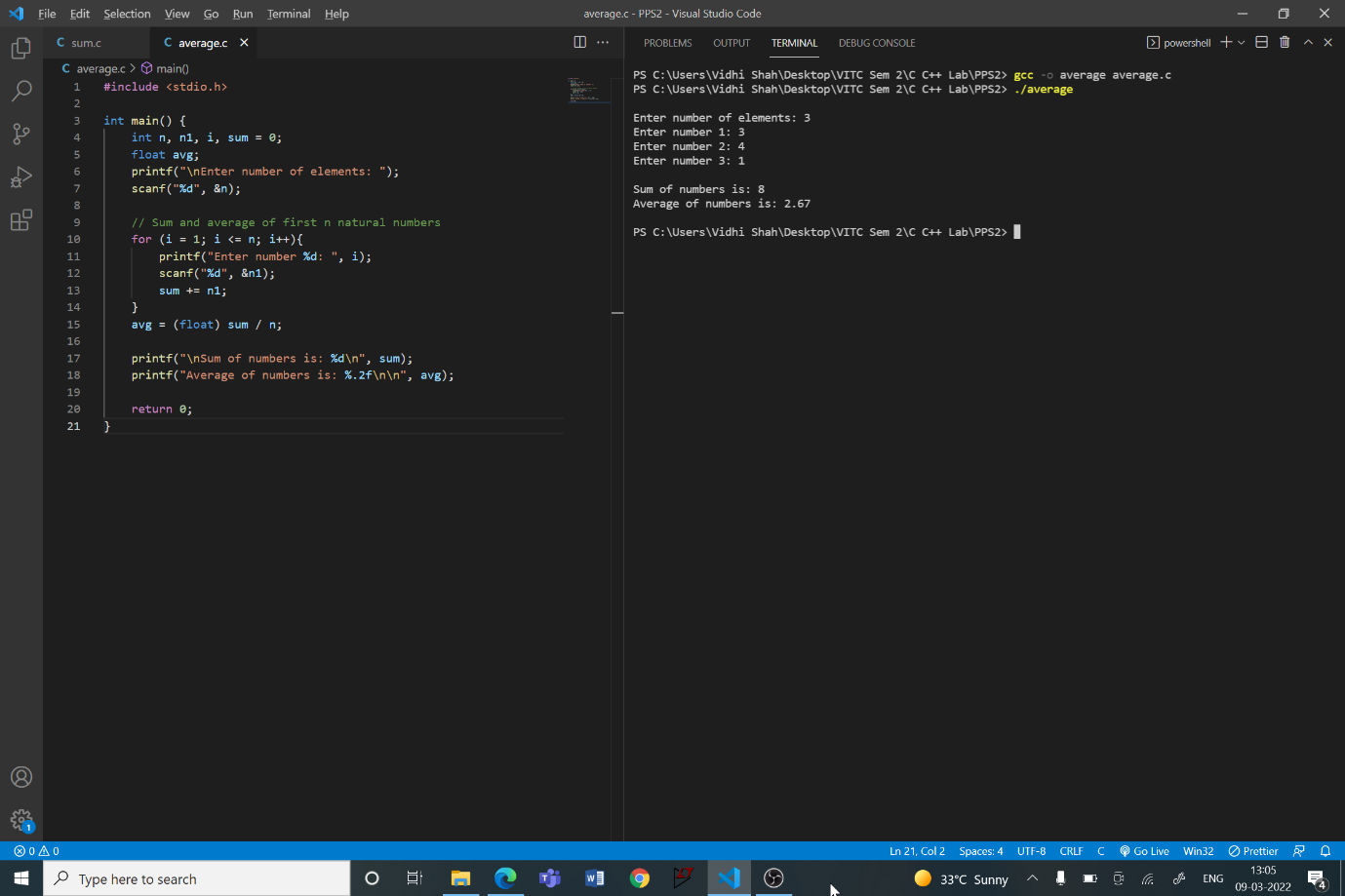
Add the number to sum

Increment i by 1

Step 3: Average = Sum / Number of elements

Step 4: Print sum and average of the numbers in separate line

**Code:**

****

**Q3**

**Aim:**

Write a program in C using for loops to display the pattern like right angle triangle using an asterisk.

**Procedure:**

**Input:**

Height of the triangle, h

**Output:**

Right angled triangle of height ‘h’ using asterisks

**Algorithm:**

Step 1: Read h

Step 2: Use for loop until i is less than h. Initialise i to 1. For each iteration:

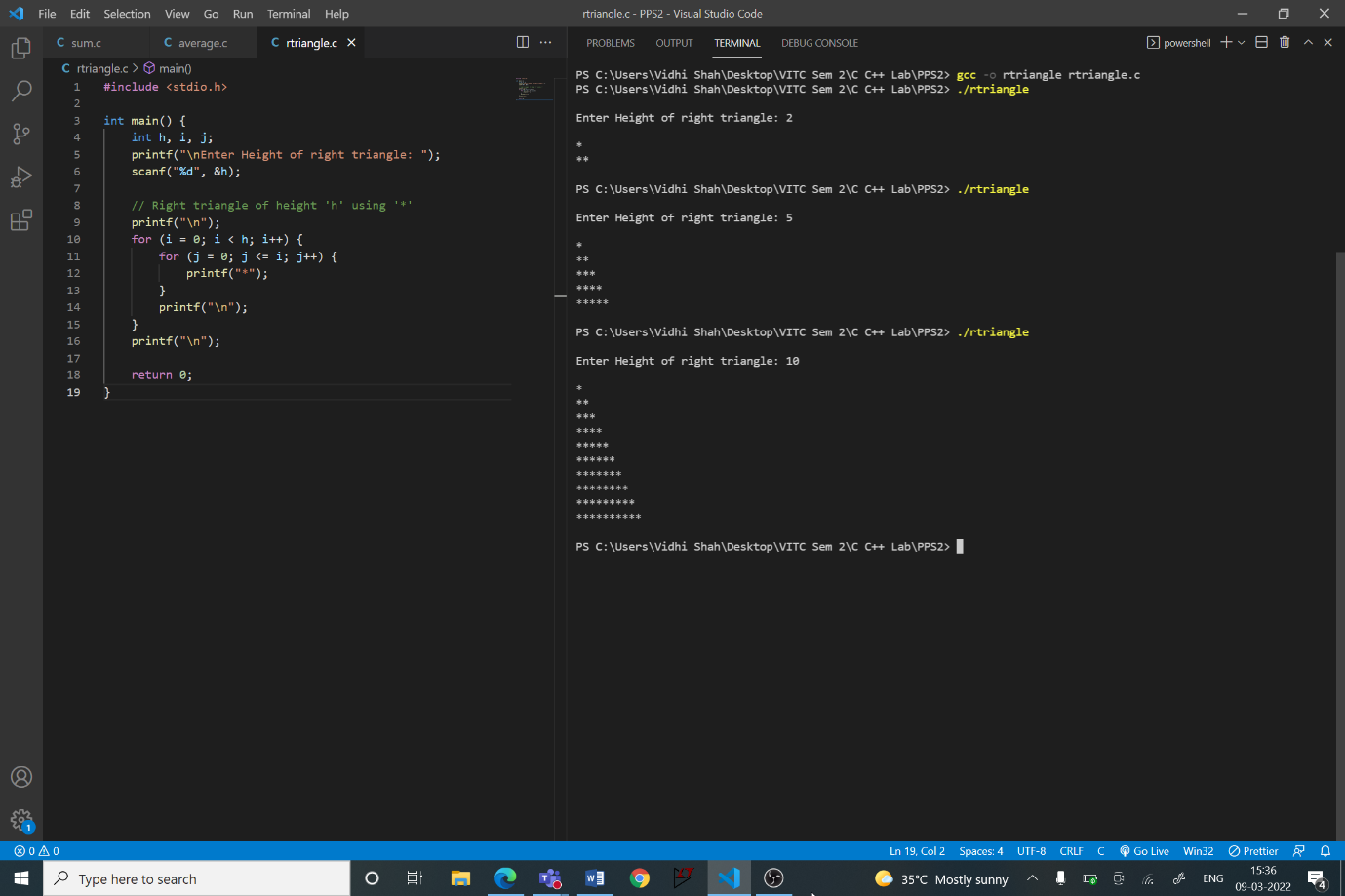
Step A: Use for loop until j is less than equal to i. Initialise j to 0. Step i: Print ‘\*’ symbol

Step ii: Increment j

Step B: Print a new line

Step C: Increment i

**Code:**

****

**PPS3**

**Q1**

**Aim:**

Write a program in C to print the pattern:

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

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

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

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

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

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

**Procedure:**

**Input:**

No input

**Output:**

Above Pattern

**Algorithm:**

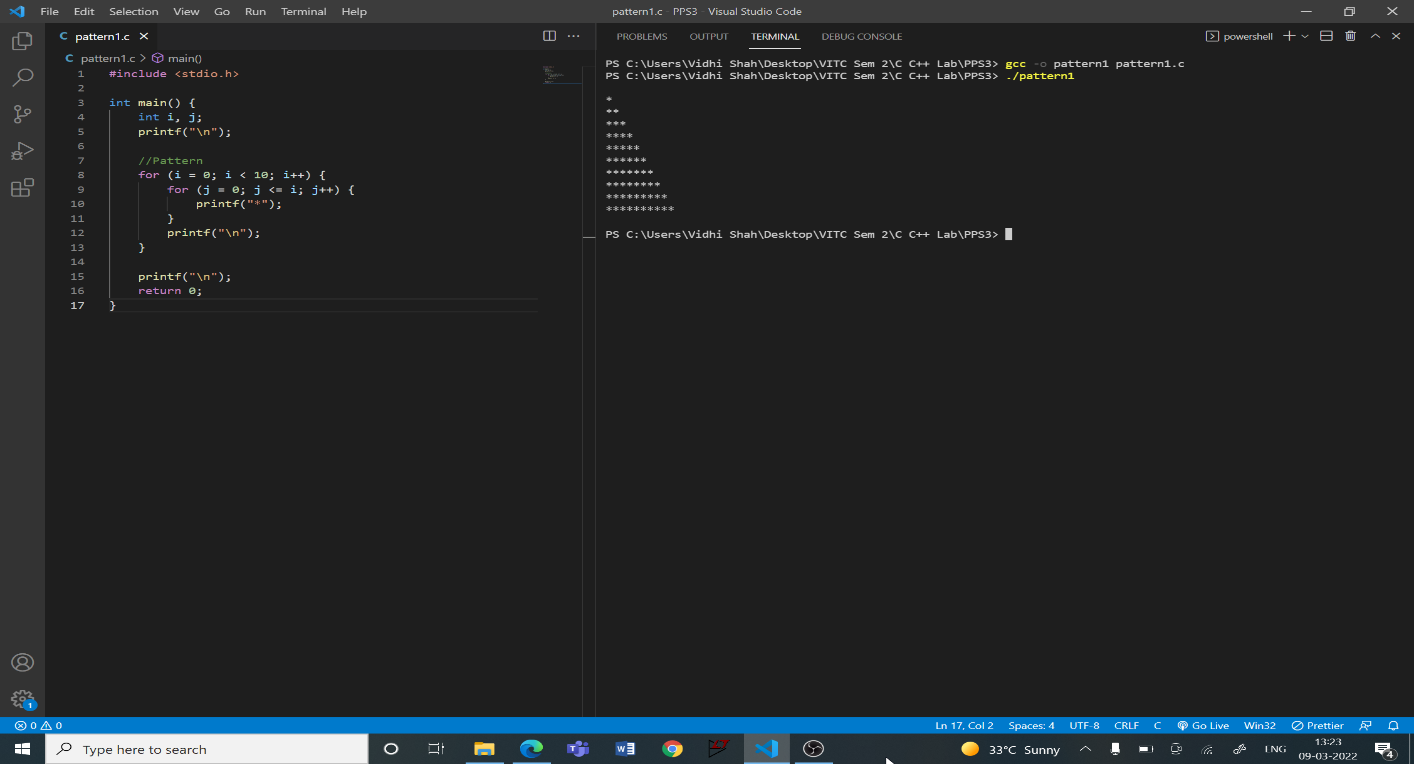
Step 1: Use for loop until i is less than 10. Initialise i to 1. For each iteration:

Step A: Use for loop until j is less than equal to i. Initialise j to 0. Step i: Print ‘\*’ symbol

Step ii: Increment j

Step B: Print a new line

Step C: Increment i

**Code:**

**Q2**

**Aim:**

Write a program in C to print the pattern:

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

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

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

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

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

\*\*\*\*\*

\*\*\*\*

\*\*\*

\*\*

\*

**Procedure:**

**Input:**

No input

**Output:**

Above Pattern

**Algorithm:**

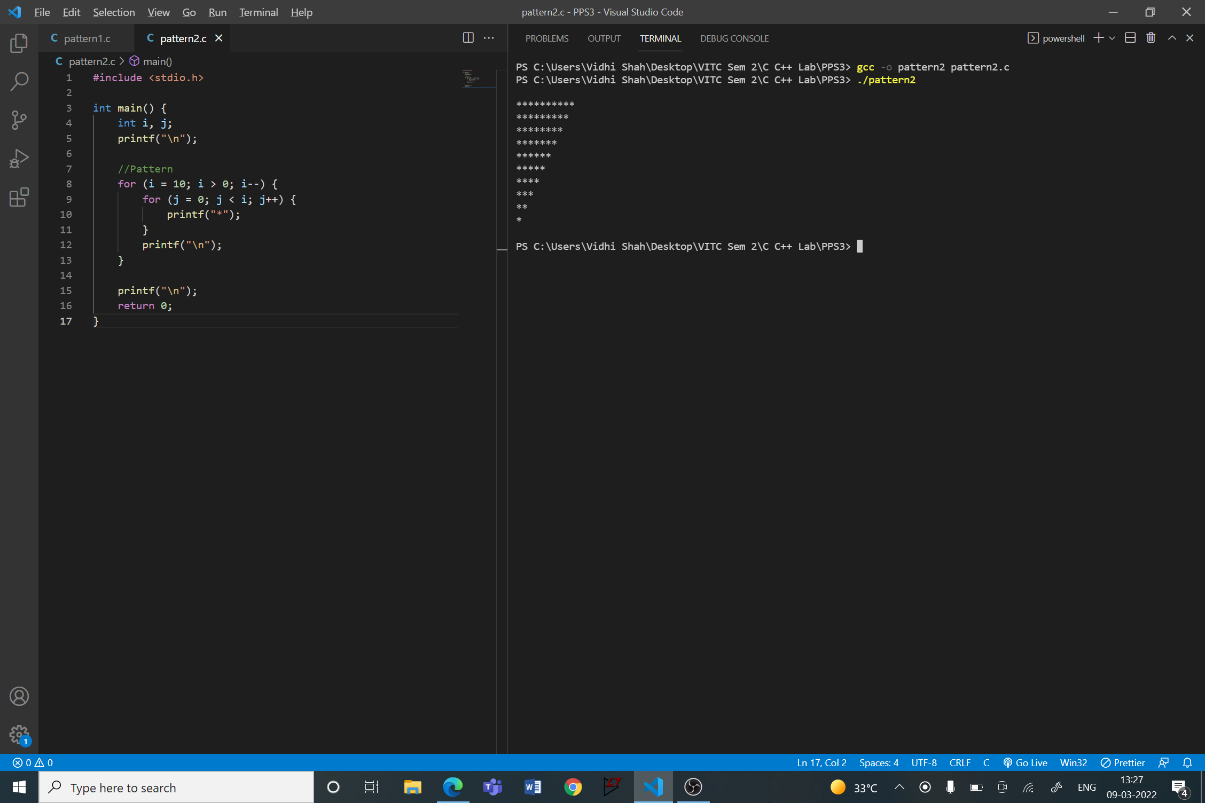
Step 1: Use for loop until i is greater than 0. Initialise i to 1. For each iteration:

Step A: Use for loop until j is less than i. Initialise j to 0. Step i: Print ‘\*’ symbol

Step ii: Increment j

Step B: Print a new line

Step C: Increment i

**Code:**

**Q3**

**Aim:**

Write a program in C to print the pattern:

\*

\*\*\*

\*\*\*\*\*

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

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

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

\*\*\*\*\*

\*\*\*

\*

**Procedure:**

**Input:**

No input

**Output:**

Above Pattern

**Code:**

#include <stdio.h>

int main() {

    int i, j, n = 9;

    printf("\n");

    //Upper Triangle

    for (i = 1; i <= n; i = i + 2) {

        for (j = 0; j < (n-i)/2; j++) {

            printf(" ");

        }

        for (j = 0; j < i; j++) {

            printf("\*");

        }

        for (j = 0; j < (n-i)/2; j++) {

            printf(" ");

        }

        printf("\n");

    }

    //Lower Triangle

    for (i = n - 2; i > 0; i = i - 2) {

        for (j = 0; j < (n-i)/2; j++) {

            printf(" ");

        }

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

            printf("\*");

        }

        for (j = 0; j < (n-i)/2; j++) {

            printf(" ");

        }

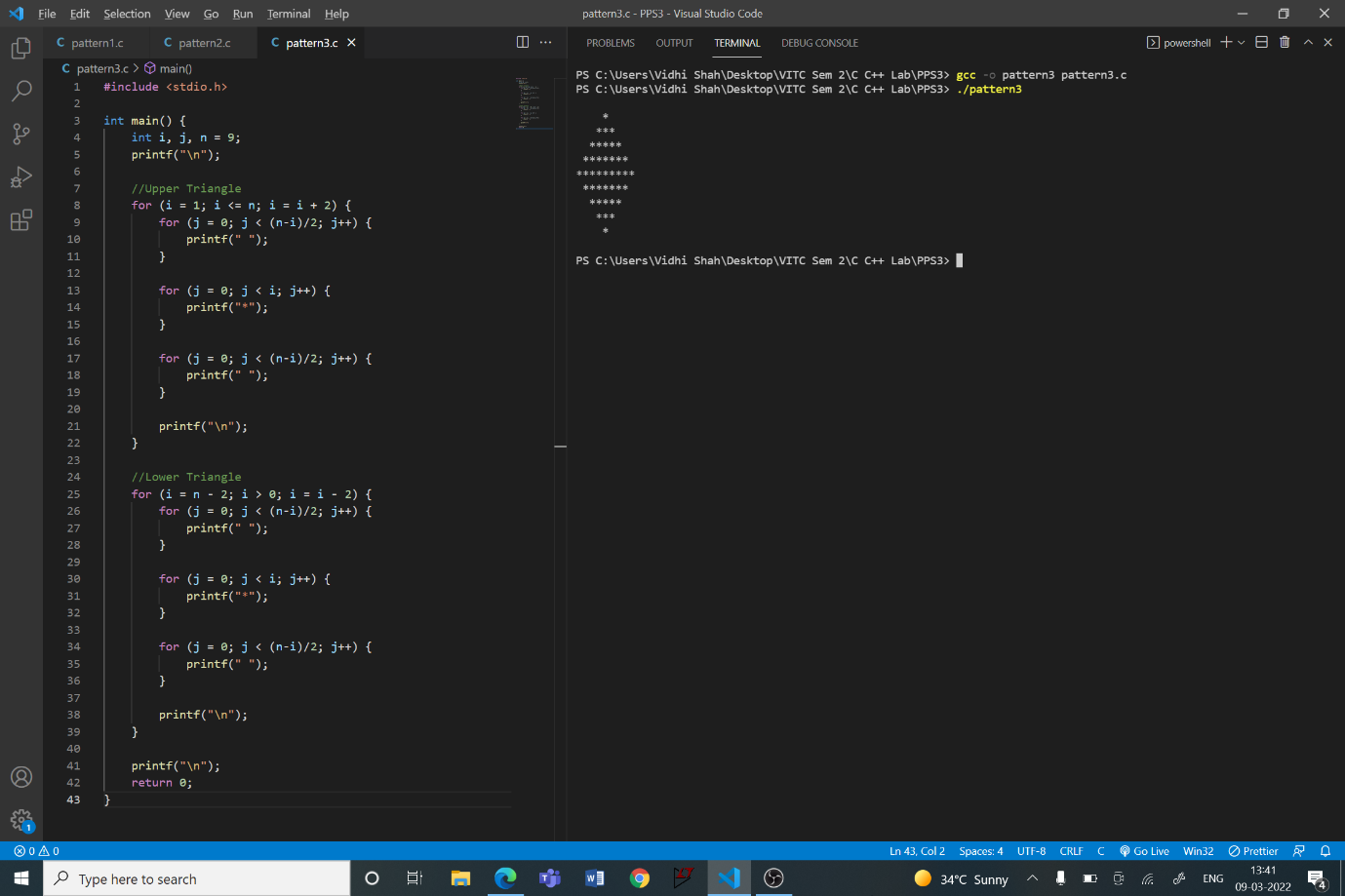
        printf("\n");

    }

    printf("\n");

    return 0;

}

****