**1. (Spiral Matrix)**

#include <stdio.h>

void spiralPrint(int matrix[4][4], int rows, int cols) {

    int top = 0, bottom = rows;

    int left = 0, right = cols;

    while (top < bottom && left < right) {

        for (int i = left; i < right; i++)

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

        top++;

        for (int i = top; i < bottom; i++)

            printf("%d ", matrix[i][right - 1]);

        right--;

        if (top < bottom) {

            for (int i = right - 1; i >= left; i--)

                printf("%d ", matrix[bottom - 1][i]);

            bottom--;

        }

        if (left < right) {

            for (int i = bottom - 1; i >= top; i--)

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

            left++;

        }

    }

}

int main() {

    int matrix[4][4] = {

        {1, 2, 3, 4},

        {5, 6, 7, 8},

        {9, 10, 11, 12},

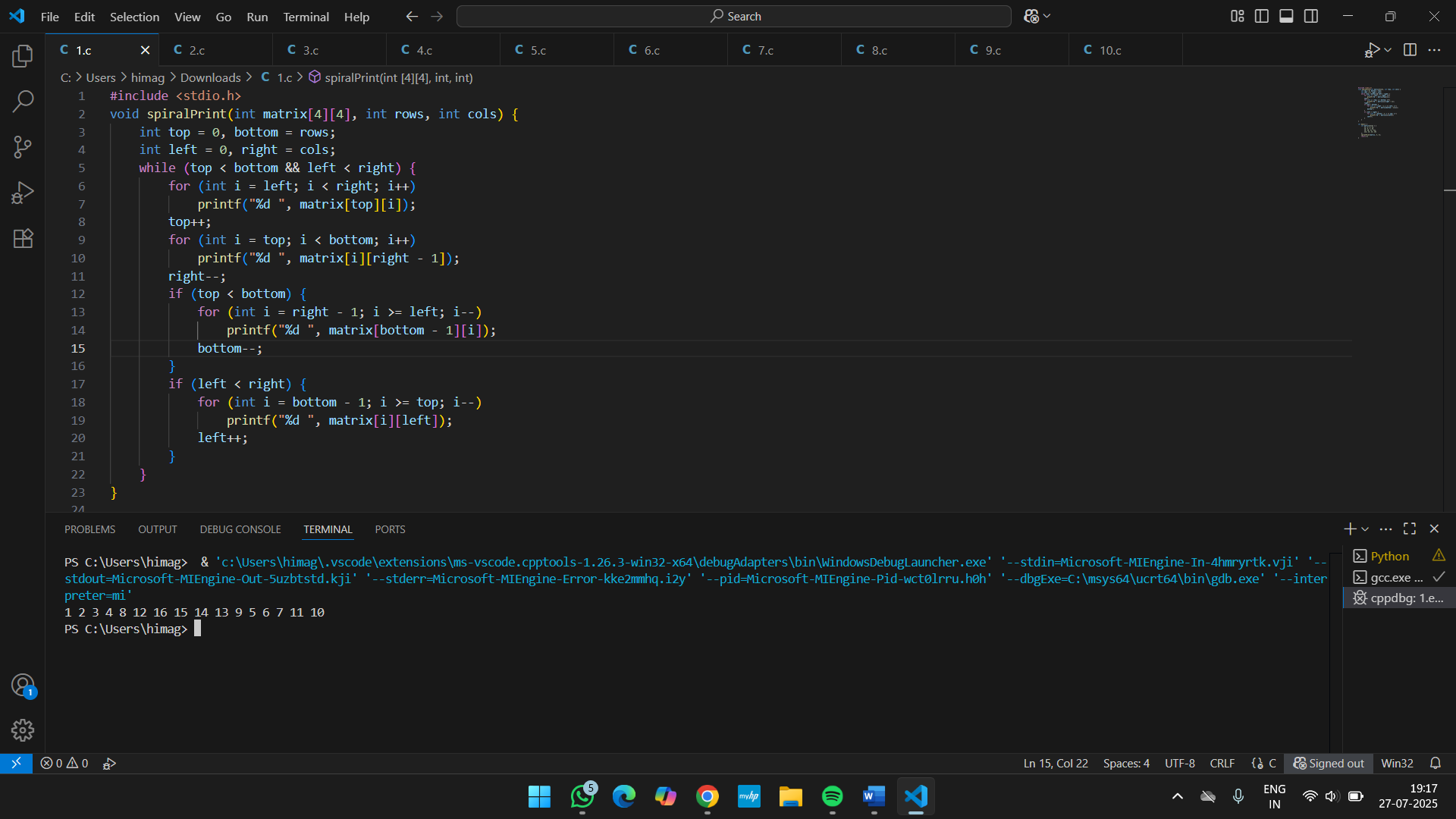
        {13, 14, 15, 16}

    };

    spiralPrint(matrix, 4, 4);

    return 0;

}



**2. (Rotate Matrix)**

#include <stdio.h>

int main(){

    int mat[3][3]={{1,2,3},{4,5,6},{7,8,9}};

    for (int i=0; i<3; i++){

        for (int j=0; j<3; j++){

            int temp=mat[i][j];

            mat[i][j]=mat[j][i];

            mat[j][i]=temp;

        }

    }

    for (int i=0; i<3; i++){

        int s=0, end=3-1;

        while (s<end){

            int temp=mat[i][s];

            mat[i][s]=mat[i][end];

            mat[i][end]=temp;

            s++;

            end--;

        }

    }

    printf("Rotated matrix : ");

    for (int i=0; i<3; i++){

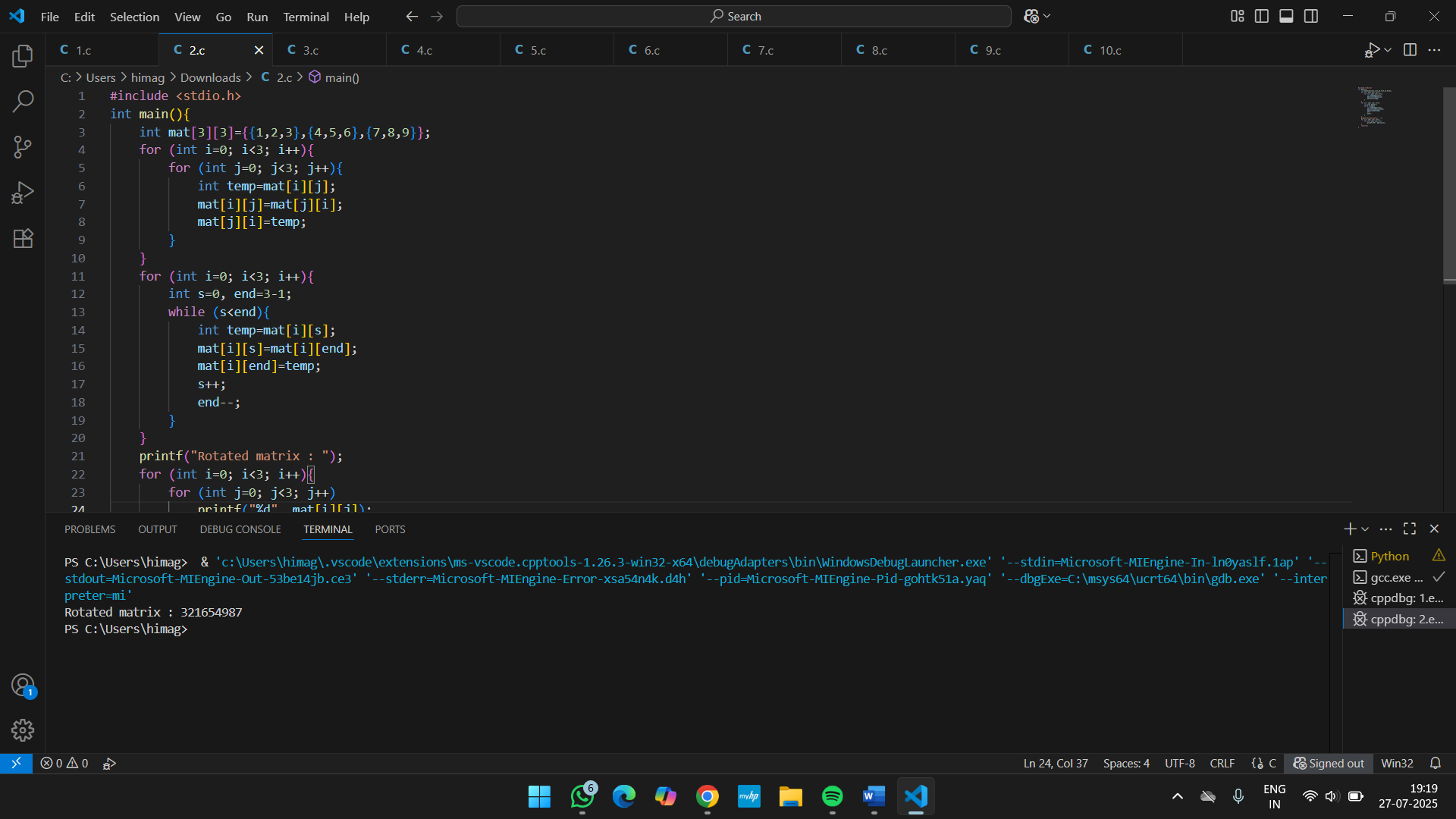
        for (int j=0; j<3; j++)

            printf("%d", mat[i][j]);

    }

    return 0;

}



**3. (Sum of Diagonals)**

#include <stdio.h>

int main(){

    int mat[3][3]={{1,2,3},{4,5,6},{7,8,9}};

    int s1=0, s2=0;

    for (int i=0; i<3; i++){

        for (int j=0; j<3; j++){

            if (i==j)

                s1+=mat[i][j];

            if ((i+j)==2)

                s2+=mat[i][j];

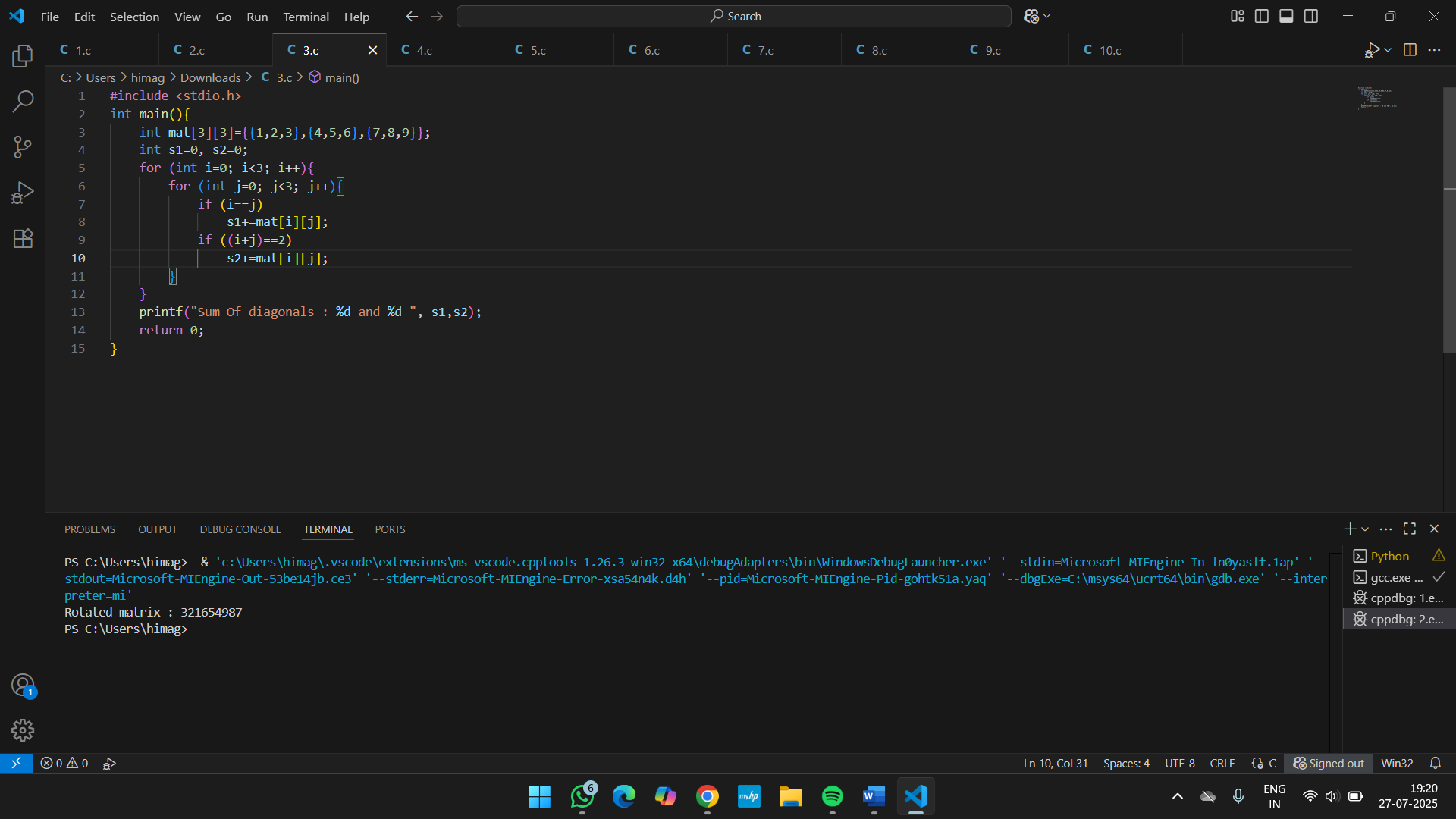
        }

    }

    printf("Sum Of diagonals : %d and %d ", s1,s2);

    return 0;

}



**4. (Transpose of a matrix)**

#include <stdio.h>

int main(){

    int mat[3][3]={{1,2,3},{4,5,6},{7,8,9}};

    for (int i=0; i<3; i++){

        for (int j=0; j<3; j++){

            int temp=mat[i][j];

            mat[i][j]=mat[j][i];

            mat[j][i]=temp;

        }

    }

    printf("Transpose : \n");

    for (int i=0; i<3; i++){

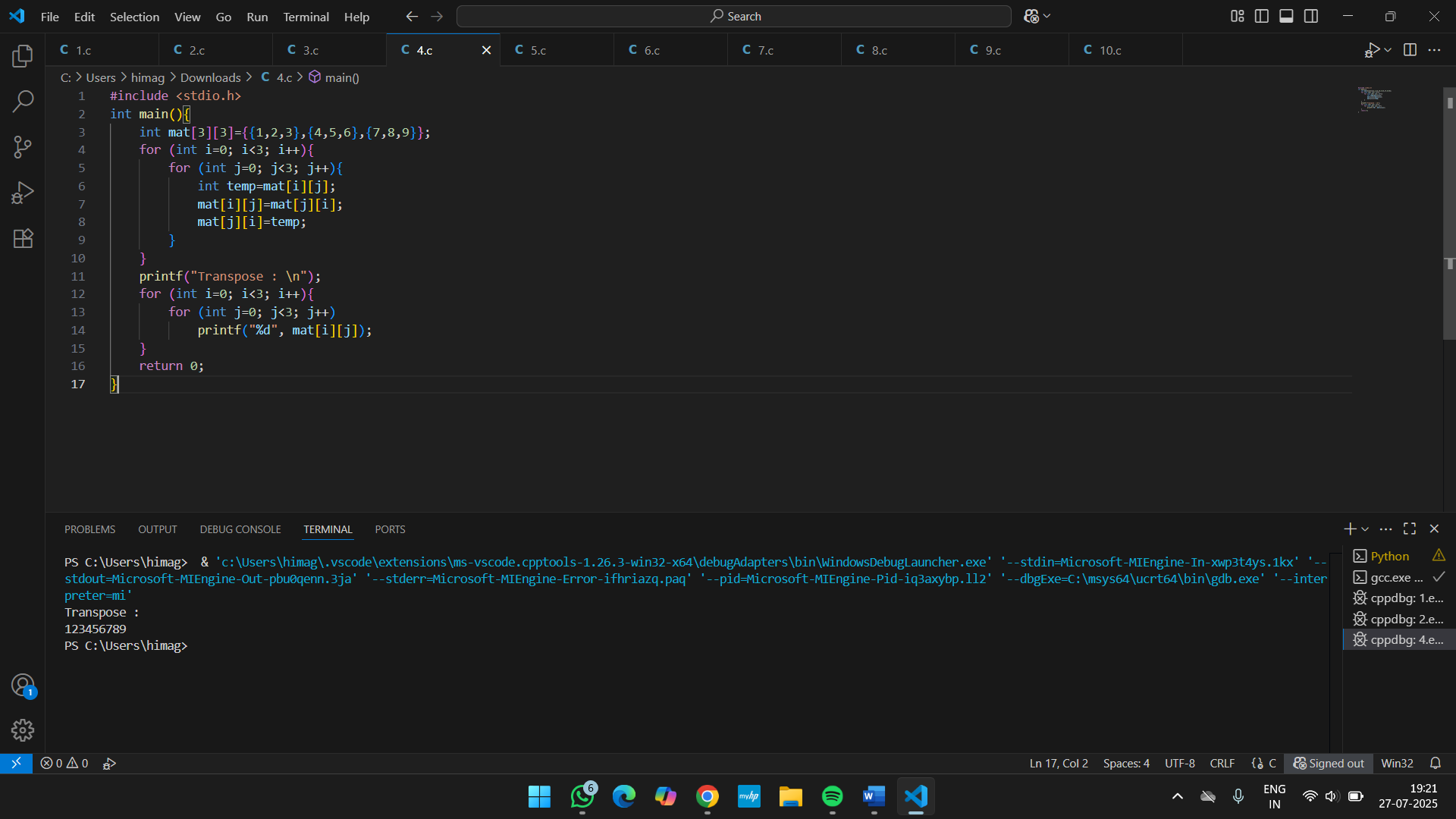
        for (int j=0; j<3; j++)

            printf("%d", mat[i][j]);

    }

    return 0;

}



**5. (Sparse Matrix)**

#include <stdio.h>

int main(){

    int mat[3][3]={{1,0,0},{0,0,0},{0,1,0}};

    int m=3, n=3, c=0;

    for (int i=0; i<m; i++){

        for (int j=0; j<n; j++){

            if (mat[i][j]==0)

                c++;

        }

    }

    if (c>(m\*n)/2)

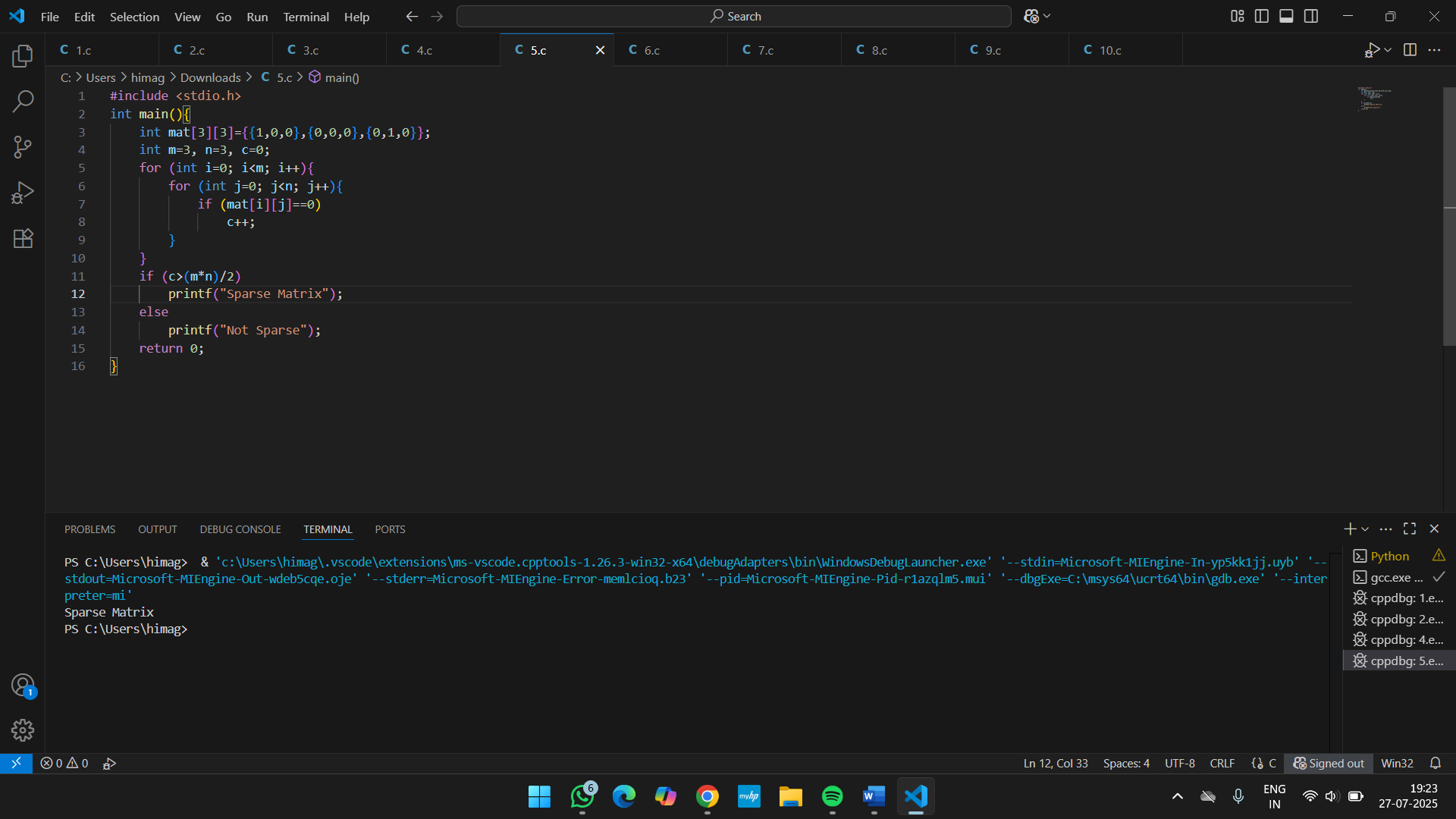
        printf("Sparse Matrix");

    else

        printf("Not Sparse");

    return 0;

}



**6. (Add 2 polynomials)**

#include <stdio.h>

void printPolynomial(int poly[], int size) {

    for (int i = 0; i < size; i++) {

        if (poly[i] == 0) continue;

        if (i != 0 && poly[i] > 0)

            printf("+ ");

        printf("%dx^%d ", poly[i], size - i - 1);

    }

    printf("\n");

}

int main() {

    int poly1[] = {3, 4, 5};

    int poly2[] = {1, 2, 3};

    int size1 = sizeof(poly1) / sizeof(poly1[0]);

    int size2 = sizeof(poly2) / sizeof(poly2[0]);

    int maxSize = size1 > size2 ? size1 : size2;

    int result[maxSize];

    for (int i = 0; i < maxSize; i++)

        result[i] = 0;

    for (int i = 0; i < size1; i++)

        result[maxSize - size1 + i] += poly1[i];

    for (int i = 0; i < size2; i++)

        result[maxSize - size2 + i] += poly2[i];

    printf("Polynomial 1: ");

    printPolynomial(poly1, size1);

    printf("Polynomial 2: ");

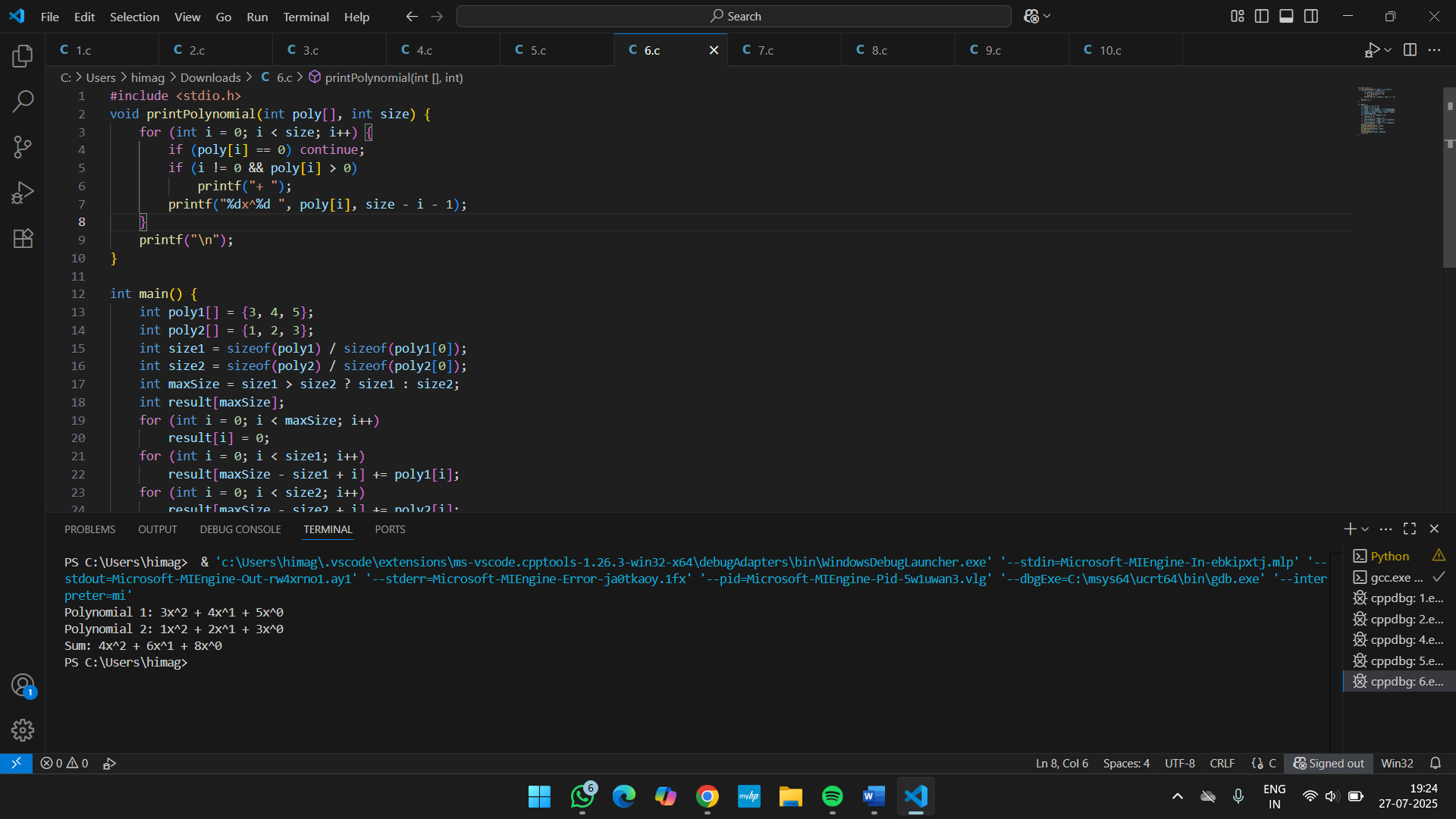
    printPolynomial(poly2, size2);

    printf("Sum: ");

    printPolynomial(result, maxSize);

    return 0;

}



**7. (Multiply 2 Polynomials)**

#include <stdio.h>

int main(){

    int a[]={5,0,1,6};

    int b[]={1,2,4};

    int m=sizeof(a)/sizeof(a[0]);

    int n=sizeof(b)/sizeof(b[0]);

    int s[m+n-1];

    for (int i=0; i<m+n-1; i++)

        s[i]=0;

    for (int i=0; i<m; i++){

        for (int j=0; j<n; j++)

            s[i+j]=a[i]\*b[j];

    }

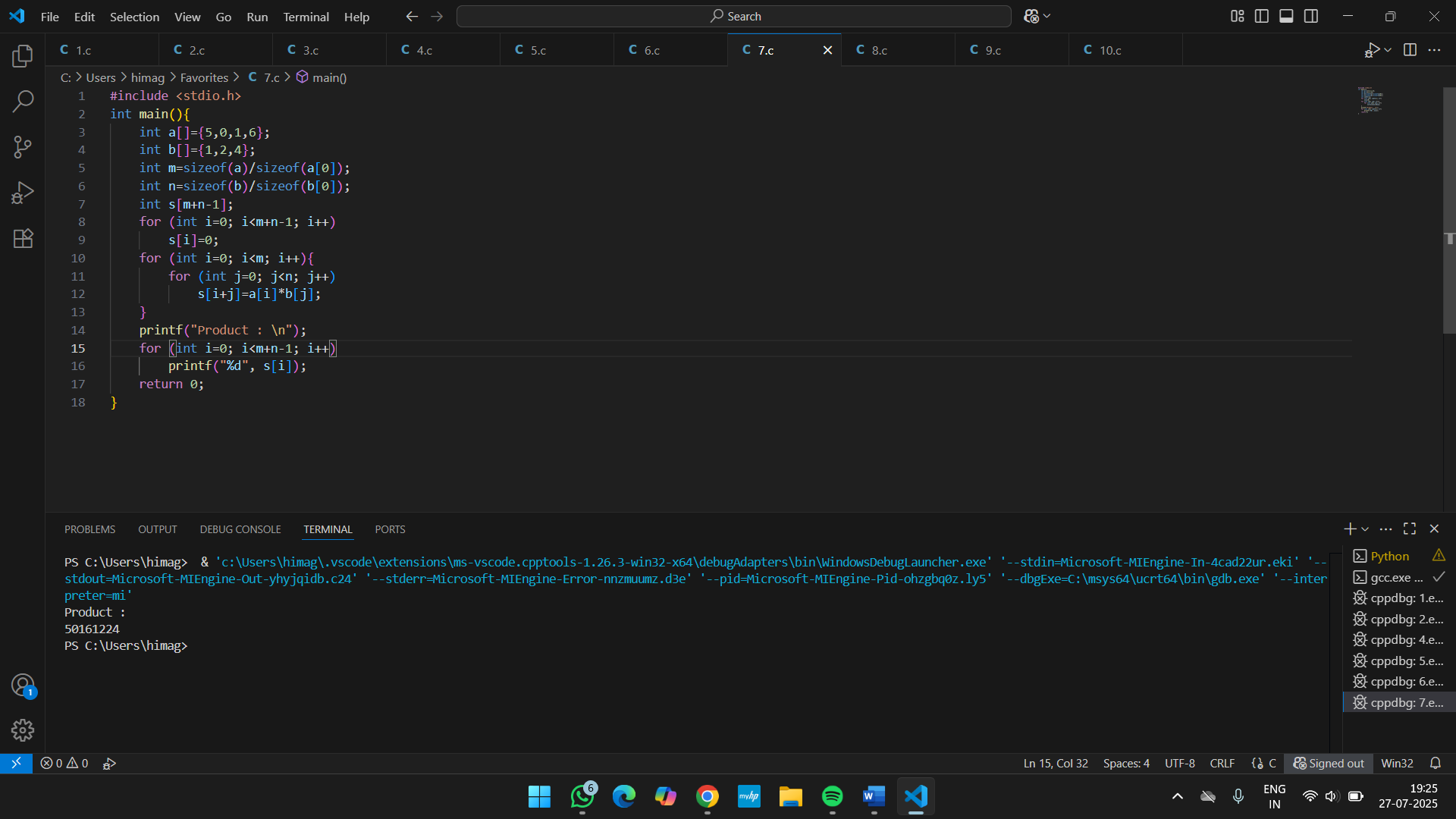
    printf("Product : \n");

    for (int i=0; i<m+n-1; i++)

        printf("%d", s[i]);

    return 0;

}



**9. (Identity matrix check)**

#include <stdio.h>

#include <stdbool.h>

bool isIdentityMatrix(int mat[][4], int n) {

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            if (i == j && mat[i][j] != 1)

                return false;

            if (i != j && mat[i][j] != 0)

                return false;

        }

    }

    return true;

}

int main() {

    int mat[4][4] = {

        {1, 0, 0, 0},

        {0, 1, 0, 0},

        {0, 0, 1, 0},

        {0, 0, 0, 1}

    };

    if (isIdentityMatrix(mat, 4))

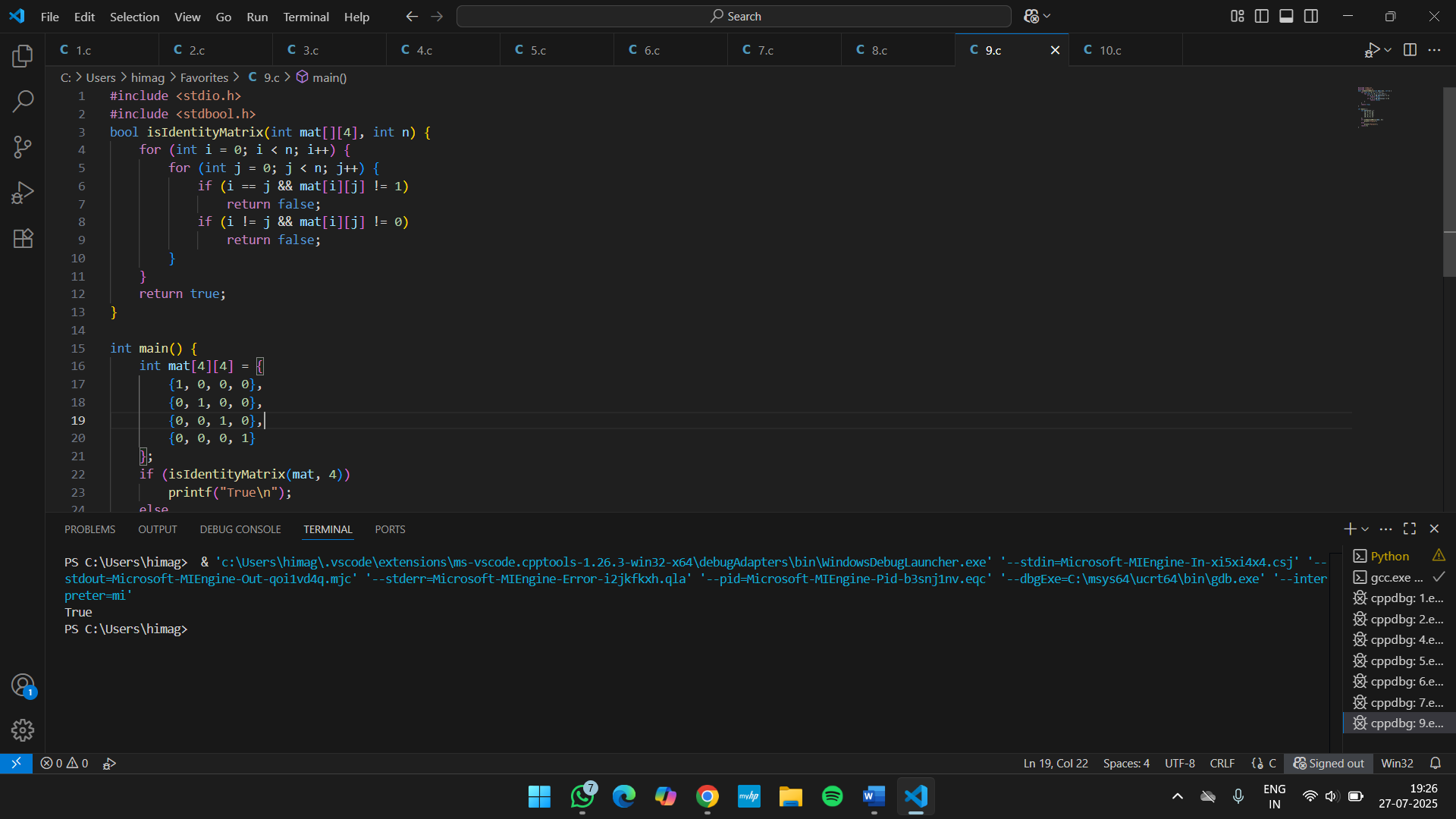
        printf("True\n");

    else

        printf("False\n");

    return 0;

}



**10. (Count zeros)**

#include <stdio.h>

#define N 5

int countZeroes(int mat[N][N])

{

    int row = N - 1, col = 0;

    int count = 0;

    while (col < N)

    {

        while (mat[row][col])

            if (--row < 0)

                return count;

        count += (row + 1);

        col++;

    }

    return count;

}

int main()

{

    int mat[N][N] =

    {

        { 0, 0, 0, 0, 1 },

        { 0, 0, 0, 1, 1 },

        { 0, 1, 1, 1, 1 },

        { 1, 1, 1, 1, 1 },

        { 1, 1, 1, 1, 1 }

    };

    printf("%d",countZeroes(mat));

    return 0;

}

