**Compressed Path Sum in Zigzag Matrix:**

Qualification Round Submission

Name: Debasis Panigrahi

Email: your-email@example.com

Date: 10th July 2025

GitHub Repo: ..

**Problem Statement:**

You are given a square matrix NxN of positive integers. You need to:

1. Traverse the matrix in a zigzag diagonal fashion:

o Start from the top-left element.

o Move diagonally up-right (↗) until you hit the boundary.

o Then move diagonally down-left (↙), and so on, alternating directions until

you reach the bottom-right.

2. At every step, maintain the running sum of visited elements.

3. Then, for every prime number found in this path, subtract it from the sum instead

of adding it.

Your task is to return the final computed value.

**Solution Approach:**

- I first implemented a function `is\_prime` to check if a number is prime without using any built-in libraries.

- Then, I used diagonal traversal logic based on the sum of indices (`i + j = d`) to alternate direction.

- For even d, I traversed ↗ (up-right), and for odd d, I traversed ↙ (down-left).

- At each visited cell, if the number is prime, I subtracted it; else, I added it to the running sum.

- Matrix elements were accessed using pointer syntax as per instructions.

**Sample Input / Output:**

Input Matrix:

1 2 3

4 5 6

7 8 9

Zigzag Path: 1 → 2 → 4 → 5 → 3 → 6 → 8 → 7 → 9

Prime Numbers: 2, 3, 5, 7

Final Sum = 1 - 2 + 4 - 5 - 3 + 6 + 8 - 7 + 9 = 11

Output:

11

**Source Code:**

#include <stdio.h>

#include <stdlib.h>

int is\_prime(int num) {

if (num <= 1) return 0;

if (num == 2) return 1;

if (num % 2 == 0) return 0;

for (int i = 3; i \* i <= num; i += 2) {

if (num % i == 0) return 0;

}

return 1;

}

int compute\_zigzag\_sum(int\*\* matrix, int n) {

int sum = 0;

for (int d = 0; d <= 2 \* (n - 1); d++)

{

if (d % 2 == 0) {

int row = (d < n) ? d : n - 1;

int col = d - row;

while (row >= 0 && col < n)

{

int val = \*(\*(matrix + row) + col);

sum += is\_prime(val) ? -val : val;

row--;

col++;

}

}

else

{

int col = (d < n) ? d : n - 1;

int row = d - col;

while (col >= 0 && row < n){

int val = \*(\*(matrix + row) + col);

sum += is\_prime(val) ? -val : val;

row++;

col--;

}

}

}

return sum;

}

int main() {

int n;

printf("Enter matrix size:: ");

scanf("%d", &n);

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

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

matrix[i] = (int\*)malloc(n \* sizeof(int));

printf("Enter the matrix values (%dx%d)::\n", n, n);

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

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

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

int result = compute\_zigzag\_sum(matrix, n);

printf("The Final Zigzag Computed Sum: %d\n", result);

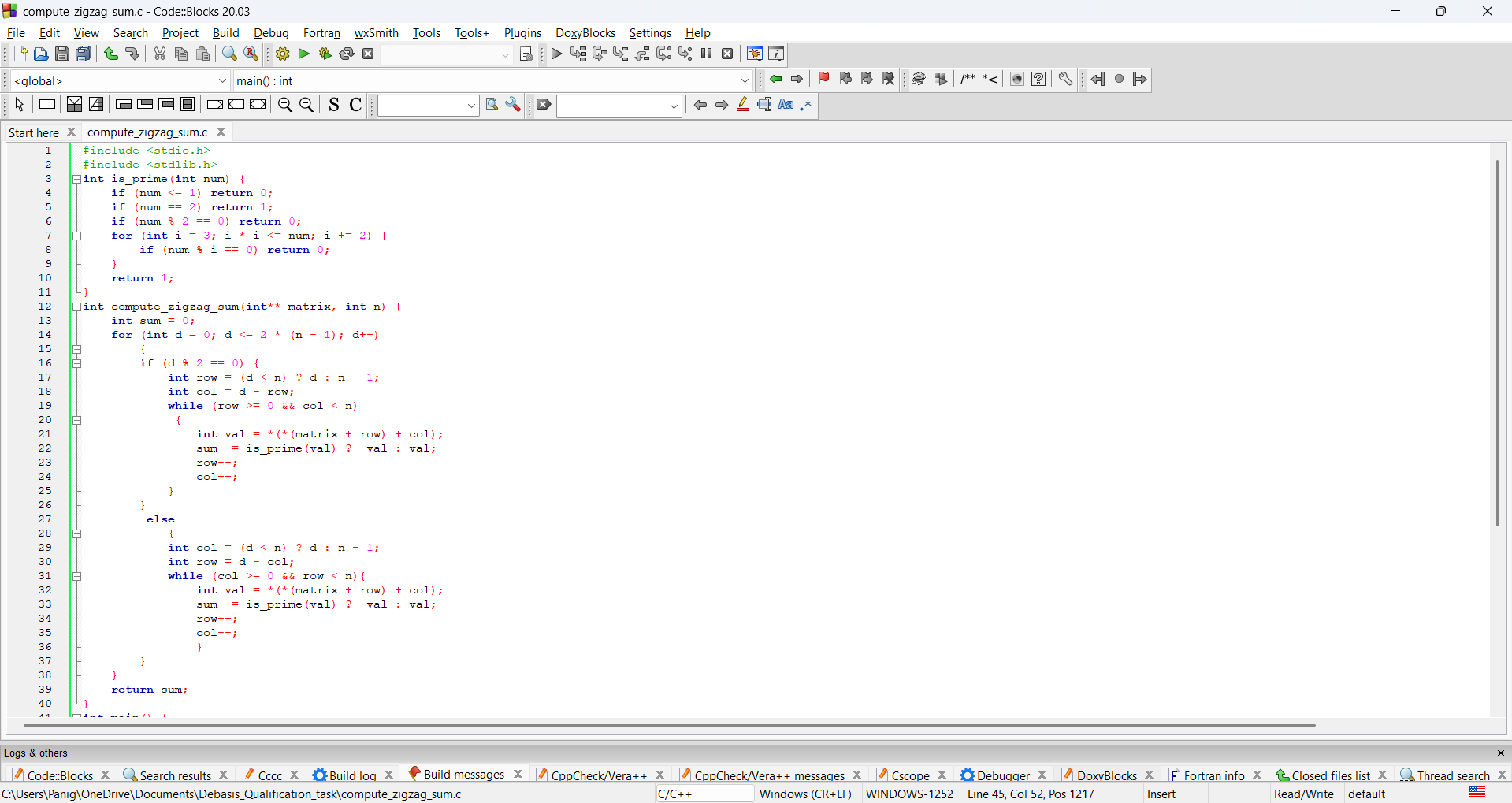
for (int i = 0; i < n; i++)

free(matrix[i]);

free(matrix)

return 0;

}



**Repository Link**: [https://github.com/Deathsythe69/compressed-zigzag-sum.git](%20https:/github.com/Deathsythe69/compressed-zigzag-sum.git)

