**Code :**

**#include <iostream>**

**using namespace std;**

**bool checkprime(int n){**

**if (n<=1) return 0; //1 & 0 are not prime number**

**if (n==2) return 1; //2 is a prime number**

**for (int i=2; i<= (n/2); i++){**

**if (n%i ==0) return 0; //not a prime hence false**

**}**

**return 1; //if the for loop is iterated completely, then its a prime number hence true**

**}**

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

**int sum=0;**

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

**//d is the diagonal number, there are total of 2\*n diagonals in a nxn matrix**

**if (d%2==0){**

**//up-right**

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

**//initialise row value to current diagonal numberor else boundary value**

**int j=d-i; //here for each diagonal, d=i+j**

**//sum of row and column number gives diagonal number**

**while(i>=0 && j<n){**

**int x= \*(\*(matrix+i)+j); //matrix[i][j]**

**//pointer access for i-th row and j-th element**

**sum += checkprime(x)? -x :x;**

**i--;**

**j++;**

**//traverse through the diagonal in upward right direction**

**}**

**}**

**else{**

**//down-left**

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

**//initialise column to current diagonal or else boundary value**

**int i=d-j; //since d=i+j**

**while(j>=0&&i<n){**

**int x= \*(\*(matrix+i)+j);**

**sum+= checkprime(x)?-x:x;**

**i++;**

**j--;**

**//traverse through the diagonal in downward left direction**

**}**

**}**

**}**

**return sum;**

**}**

**int main (){**

**int n;**

**cout<<"Enter the number of rows for square matrix : ";**

**cin>>n;**

**int \*\*matrix = new int\*[n];**

**//declare a 2d array with n rows**

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

**matrix[i]=new int[n];**

**//declare each row as an array of size n**

**}**

**cout<<"Enter "<<n<<" x "<<n<<" elements for the matrix : "<<endl;**

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

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

**cin>>\*(\*(matrix+i)+j);**

**//this will take input for matrix [i][j]**

**}**

**}**

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

**cout<<"The total sum of the elements is : "<< sum<<endl;**

**//free allocated memory space**

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

**delete[] matrix[i];**

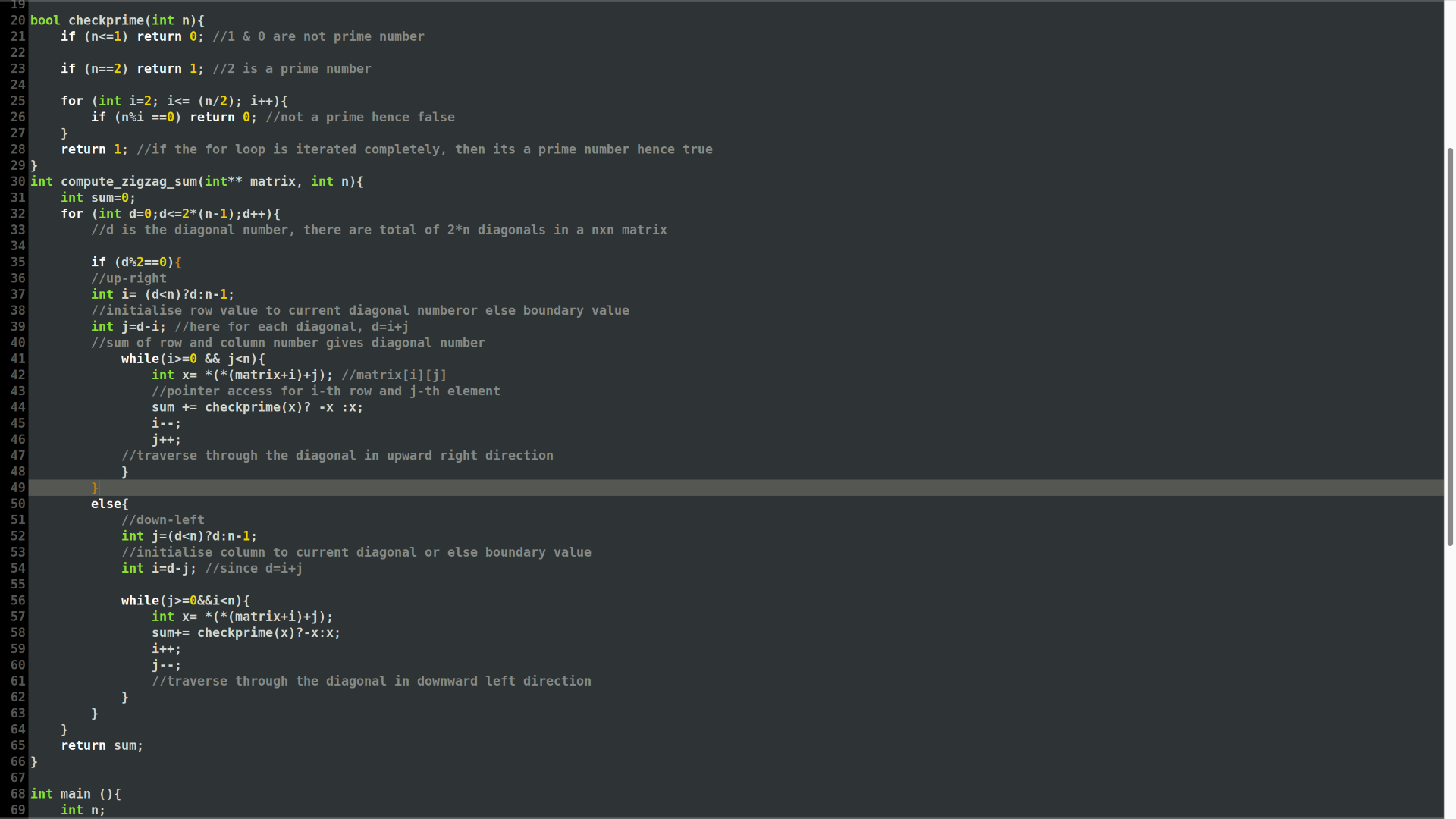
**}**

**delete [] matrix;**

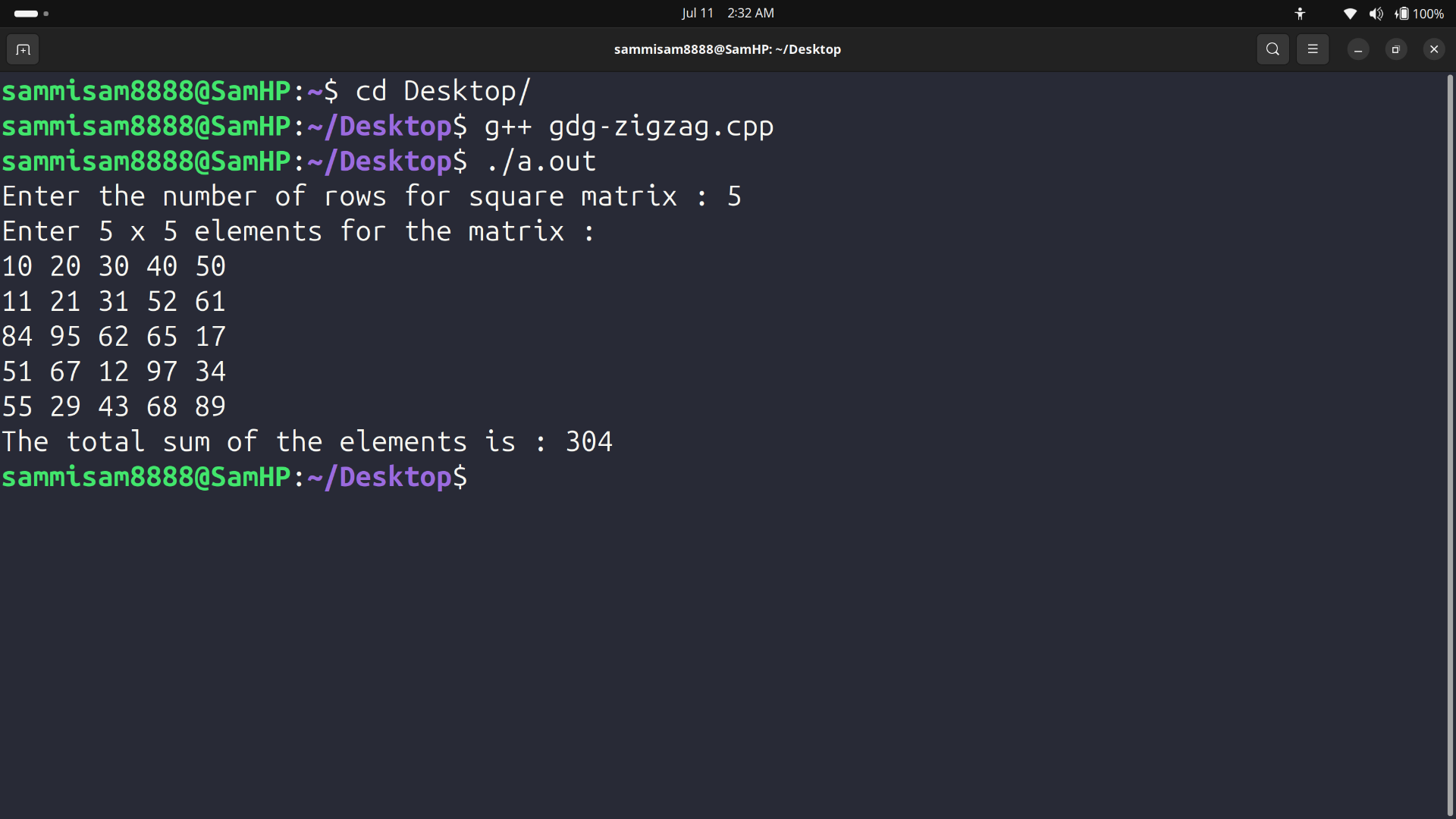
**return 0;**

**}**

**Code Screenshot :**

****

**Code Output :**

****

**Github Repository Link :** [**https://github.com/Sammisam8888/GDG-BBSR-2025-Submission**](https://github.com/Sammisam8888/GDG-BBSr-Submission)