Question 1:

Construct a C++ program engineered to determine the sum of integers within the range from 1 to a user-defined value, N. Delve into the intricacies of the calculateSum function, elucidating its utilization of a for loop to systematically add integers within the specified range.

Input format:

- A single integer N, the number till which the sum needs to be calculated.

Output format:

- Print a statement as: "The sum of all numbers 1 to N is: result", where result is the computed sum.

Title for Question 1: SUM OF N NUMBERS

Solution:

```
#include <iostream>
int calculateSum(int N) {
    int sum = 0;
    for (int i = 1; i <= N; i++) {
        sum += i;
    }
    return sum;
}
int main() {
    int N;
    std::cin >> N;
    int result = calculateSum(N);
    std::cout << "The sum of all numbers from 1 to " << N << " is: " << r
    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	5	The sum of all numbers from 1 to 5 is: 15
2	15	The sum of all numbers from 1 to 15 is: 120
3	7	The sum of all numbers from 1 to 7 is: 28
4	20	The sum of all numbers from 1 to 20 is: 210
5	43	The sum of all numbers from 1 to 43 is: 946
6	5	The sum of all numbers from 1 to 5 is: 15

White List:

Black List:

Question 2:

In a kingdom revered for its mathematical wisdom, Alex, a renowned mathematician, devised a program that could count the number of digits in any given number. Can you recreate this program to solve the enigma and count the digits of a number?

Task: Write a program that takes an integer number, n, (from -10^9 to 10^9 excluding 0) and prints the number of digits it contains. Negative numbers should be rejected, and the program should inform the user about the invalid input.

Input format:

- A single integer n.

Output format:

- If the number is negative, print: "Negative numbers are not valid inputs."
- If the number is 0, print: "The number 0 has 1 digit."
- Otherwise, print: "The number of digits in n is: result", where result is the count of digits in the number.

Title for Question 2: THE DIGIT COUNTER CHALLENGE

Solution:

```
#include <iostream>
int countDigits(int number) {
    // Handle the special case when the number is 0
    if (number == 0) {
        return 1;
    }
    int count = 0;
    // Handle negative numbers by converting them to positive
    if (number < 0) {
        number = -number;
    }


    // Count the digits using a loop
    while (number != 0) {
        number /= 10;
        count++;
    }
    return count;
}
int main() {
    int number;
    std::cin >> number;
    int digitCount = countDigits(number);
    std::cout << "The number of digits in " << number << " is: " << digit
    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	12345	The number of digits in 12345 is: 5
2	2236	The number of digits in 2236 is: 4
3	45632	The number of digits in 45632 is: 5
4	24	The number of digits in 24 is: 2
5	-983	The number of digits in -983 is: 3
6	3424	The number of digits in 3424 is: 4

White List:

Black List:

Question 3:

Write a program to generate the following series 0 2 8 14 24 34 48 62 80 98 .....

Input Format:

- The input consists of an integer 'n' which denotes the number of terms to be printed in the series.

Output Format:

- The output consists of the series and refer to the sample output for formatting.

Title for Question 3: PROBLEM STATEMENT

Solution:

```
#include<iostream>
using namespace std;
int main()
{
  int n , i , val = 0 , diff = 2 , count = 0 ;
  cin >> n ;

  for ( i = 0 ; i < n ; i++ )
  {
    if ( i == 0 )
    cout << val << " " ;
    else
    {
      val += diff ;
      cout << val << " " ;
    }
    count++ ;
    if ( count % 2 == 0 )
      diff += 4 ;
  }
```

```
  return 0 ;
}
```

TestCases:

S.No	Inputs	Outputs
1	5	0 2 8 14 24
2	8	0 2 8 14 24 34 48 62
3	5	0 2 8 14 24
4	14	0 2 8 14 24 34 48 62 80 98 120 142 168 194
5	20	0 2 8 14 24 34 48 62 80 98 120 142 168 194 224 254 288 322 360 398
6	1	0

White List:

Black List:

Question 4:

Given an array of integers, you are required to find the sum of all its elements. Implement a function or a program to achieve this.

Constraints:

- 1 ? Array length ? 10^5-10^4 ? Array elements ? 10^4

Input Format:

- The first line contains a single integer n, representing the number of elements in the array.
- The second line contains n space-separated integers which constitute the array.

Output Format:

- Output a single line saying "Sum of array elements: " followed by the sum.

Title for Question 4: Sum of elements in an array

Solution:

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;

    // Check if n is negative
```

```
    if (n < 0) {
        cout << "Invalid Input" << endl;
        return 0; // Exit the program
    }

    int *arr = new int[n];  // Dynamically allocated array

    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    int sum = 0;  // Initialize sum as 0

    for (int i = 0; i < n; i++) {
        sum += arr[i];  // Add each element to the sum
    }

    cout << "The sum of all elements in the array is: " << sum << endl;

    delete[] arr;

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	5 1 2 3 4 5	The sum of all elements in the array is: 15
2	-3 100 200 300	Invalid Input
3	3 -10 -20 -30	The sum of all elements in the array is: -60
4	1 2345678	The sum of all elements in the array is: 2345678
5	6 1 9 2 8 3 7	The sum of all elements in the array is: 30
6	-2 -8 -7	Invalid Input

White List:

Black List:

Question 5:

Implement a program that reads a square matrix from the user and finds the sum of its main diagonal elements. A matrix is defined as a 2D array of numbers. The main diagonal of a square matrix is the diagonal from the top left to the bottom right. Your program should take dynamic input for the size of the matrix and its elements.

Input Format:

- An integer n for the number of rows and columns of the square matrix.
- n * n integers separated by space, representing the elements of the matrix, row by row.

Output Format:

- Output the sum of the main diagonal elements.

Title for Question 5: Sum of diagonals

Solution:

```
#include <iostream>

using namespace std;

int main() {
    int n;

    // Input for the square matrix size
    //cout << "Enter the number of rows and columns for the square matrix
    cin >> n;

    // Dynamic allocation of 2D array
    int** matrix = new int*[n];
    for(int i = 0; i < n; i++) {
        matrix[i] = new int[n];
    }

    // Input for matrix elements
    //cout << "Enter the elements of the matrix:" << endl;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            cin >> matrix[i][j];
        }
    }

    // Calculate the sum of main diagonal elements
    int sum = 0;
    for (int i = 0; i < n; ++i) {
        sum += matrix[i][i];
    }

    // Output the sum
    cout << "The sum of the main diagonal elements is: " << sum << endl;

    // Freeing dynamically allocated memory
    for(int i = 0; i < n; i++) {
        delete[] matrix[i];
    }
    delete[] matrix;

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs

1	3 1 2 3 4 5 6 7 8 9	The sum of the main diagonal elements is: 15
2	2 -1 2 3 -4	The sum of the main diagonal elements is: -5
3	2 1 2 3 4	The sum of the main diagonal elements is: 5
4	3 1 2 3 4 5 6 7 8 9	The sum of the main diagonal elements is: 15
5	4 -1 -2 -3 -4 -5 -6 -7 -8 -9 -10 -11 -12 -13 -14 -15 -16	The sum of the main diagonal elements is: -34
6	1 100	The sum of the main diagonal elements is: 100

White List:

Black List:

Question 6:

Write a program that reads two matrices from the user and multiplies them. A matrix is defined as a 2D array of numbers. You can only multiply matrices if the number of columns in the first matrix is equal to the number of rows in the second matrix.

Your program should take dynamic input for the size of the matrices and their elements.

Input Format :

- Two integers n1 and m1 for the number of rows and columns of the first matrix.
- n1 * m1 integers separated by space, representing the elements of the first matrix, row by row.
- Two integers n2 and m2 for the number of rows and columns of the second matrix.
- n2 * m2 integers separated by space, representing the elements of the second matrix, row by row.

Output Format:

- Output the resulting matrix from the multiplication, element by element, row by row.

Title for Question 6: Matrix multiplication

Solution:

```
#include <iostream>

using namespace std;

int main() {
    int n1, m1, n2, m2;

    // Input for the first matrix size
```

```
 //   cout << "Enter the number of rows for the first matrix: ";
    cin >> n1;
  //  cout << "Enter the number of columns for the first matrix: ";
    cin >> m1;

    // Dynamic allocation of 2D array for first matrix
    int** matrix1 = new int*[n1];
    for(int i = 0; i < n1; i++) {
        matrix1[i] = new int[m1];
    }

    // Input for first matrix elements
//    cout << "Enter the elements of the first matrix:" << endl;
    for (int i = 0; i < n1; ++i) {
        for (int j = 0; j < m1; ++j) {
            cin >> matrix1[i][j];
        }
    }

    // Input for the second matrix size
   // cout << "Enter the number of rows for the second matrix: ";
    cin >> n2;
  //  cout << "Enter the number of columns for the second matrix: ";
    cin >> m2;

    // Check for multiplication possibility
    if (m1 != n2) {
        cout << "Matrix multiplication is not possible." << endl;
        return 0;
    }

    // Dynamic allocation of 2D array for second matrix
    int** matrix2 = new int*[n2];
    for(int i = 0; i < n2; i++) {
        matrix2[i] = new int[m2];
    }

    // Input for second matrix elements
    //cout << "Enter the elements of the second matrix:" << endl;
    for (int i = 0; i < n2; ++i) {
        for (int j = 0; j < m2; ++j) {
            cin >> matrix2[i][j];
        }
    }

    // Dynamic allocation of 2D array for result matrix
    int** result = new int*[n1];
    for(int i = 0; i < n1; i++) {
        result[i] = new int[m2];
    }

    // Matrix multiplication
    for (int i = 0; i < n1; ++i) {
        for (int j = 0; j < m2; ++j) {
            result[i][j] = 0;
            for (int k = 0; k < m1; ++k) {
                result[i][j] += matrix1[i][k] * matrix2[k][j];
            }
        }
    }
```

```
    // Output the result
    cout << "The resulting matrix is:" << endl;
    for (int i = 0; i < n1; ++i) {
        for (int j = 0; j < m2; ++j) {
            cout << result[i][j] << " ";
        }
        cout << endl;
    }

    // Freeing dynamically allocated memory
    for(int i = 0; i < n1; i++) {
        delete[] matrix1[i];
    }
    delete[] matrix1;

    for(int i = 0; i < n2; i++) {
        delete[] matrix2[i];
    }
    delete[] matrix2;

    for(int i = 0; i < n1; i++) {
        delete[] result[i];
    }
    delete[] result;

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	3 3 1 0 0 0 1 0 0 0 1 3 3 1 1 1 1 1 1 1 1 1	The resulting matrix is: 1 1 1 1 1 1 1 1 1
2	2 3 1 2 3 4 5 6 2 2 1 1 1 1	Matrix multiplication is not possible.
3	2 4 1 2 3 4 5 6 7 8 4 2 1 0 0 1 1 0 0 1	The resulting matrix is: 4 6 12 14
4	2 2 -1 2 3 -4 2 2 5 6 7 8	The resulting matrix is: 9 10 -13 -14
5	2 3 1 1 1 0 0 0 3 1 1 1 1	The resulting matrix is: 3 0
6	3 3 1 2 3 4 5 6 7 8 9 3 3 9 8 7 6 5 4 3 2 1	The resulting matrix is: 30 24 18 84 69 54 138 114 90

White List:

Black List:
