Question 1:

You are tasked with writing a C++ program to calculate the factorial of a positive integer. Follow these steps:

- Prompt the user to enter a positive integer.
- Ensure that the input is a non-negative integer; if not, display an error message.
- Calculate the factorial of the input integer using an inline function.
- Display the calculated factorial.
- Implement the program according to the provided problem statement.

Function Header

```
inline unsigned long long calculateFactorial(int n)
```

Input Format

- The program prompts the user to enter a positive integer.

Output Format

- The program displays the factorial of the input integer in the following format:
- "Factorial of <input integer> is <calculated factorial>"
- when input integer is negative number it displayInvalid input. Please enter a positive integer.

Title for Question 1: Factorial Number

Solution:

```
#include <iostream>
// Inline function to calculate factorial
inline unsigned long long calculateFactorial(int n) {
    if (n == 0 || n == 1) {
        return 1;
    } else {
        unsigned long long result = 1;
        for (int i = 2; i <= n; ++i) {
            result *= i;
        }
        return result;
    }
}
int main() {
    int num;
    // Input Format: Prompt the user to enter a positive integer
//    std::cout << "Enter a positive integer: ";
```

```
    std::cin >> num;
    // Input validation
    if (num < 0) {
        std::cerr << "Invalid input. Please enter a positive integer." <<
        return 0; // Return an error code
    }
    // Calculate factorial using inline function
    unsigned long long factorial = calculateFactorial(num);
    // Output Format: Display the factorial
    std::cout << "Factorial of " << num << " is " << factorial << std::en
    return 0; // Return success
}
```

TestCases:

S.No	Inputs	Outputs
1	10	Factorial of 10 is 3628800
2	-5	Invalid input. Please enter a positive integer.
3	20	Factorial of 20 is 2432902008176640000
4	0	Factorial of 0 is 1
5	5	Factorial of 5 is 120
6	1	Factorial of 1 is 1

White List: inline unsigned long long calculateFactorial(int n)

Black List:

Question 2:

You are tasked with implementing a C++ program for matrix multiplication. The program should ask the user to enter two matrices, perform the multiplication, and display the result. Implement the matrix multiplication using an inline function. Matrix multiplication is defined as follows: Given two matrices A (m x n) and B (n x p), the product C (m x p) is calculated as C = A * B, where each element C[i][j] is the sum of the products of elements from row i of matrix A and column j of matrix B.

Your program should handle matrices of dimensions up to 10 x 10.

Input Format:

- Prompt the user to enter the dimensions (rows and columns) of matrix A and matrix B.
- Prompt the user to enter the elements of matrix A and matrix B.

Output Format:

- Display the resulting matrix C after multiplication.

Title for Question 2: Matrix Multiplication

Solution:

```
#include <iostream>
const int MAX_ROWS = 10;
const int MAX_COLS = 10;
// Inline function to perform matrix multiplication
inline void multiplyMatrices(int A[][MAX_COLS], int B[][MAX_COLS], int C[
    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < p; ++j) {
            C[i][j] = 0;
            for (int k = 0; k < n; ++k) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}
int main() {
    int A[MAX_ROWS][MAX_COLS];
    int B[MAX_ROWS][MAX_COLS];
    int C[MAX_ROWS][MAX_COLS];
    int m, n, p,z;
    // Input: Dimensions of matrix A
   // std::cout << "Enter the dimensions of matrix A (m x n):" << std::en
 //   std::cout << "Rows (m): ";
    std::cin >> m;
  //  std::cout << "Columns (n): ";
    std::cin >> n;
    // Input: Elements of matrix A
 //   std::cout << "Enter the elements of matrix A:" << std::endl;
    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < n; ++j) {
        //    std::cout << "A[" << i + 1 << "][" << j + 1 << "]: ";
            std::cin >> A[i][j];
        }
    }

    std::cin >> z;
    std::cin >> p;

    for (int i = 0; i < z; ++i) {
        for (int j = 0; j < p; ++j) {
            std::cin >> B[i][j];
        }
    }
    // Check if matrix multiplication is valid
    if (n != z) {
        std::cout << "Matrix multiplication is not valid. Number of colum
exit(0);    }
    // Perform matrix multiplication using the inline function
    multiplyMatrices(A, B, C, m, n, p);
    // Output: Display the resulting matrix C
    std::cout << "Resulting matrix C (" << m << " x " << p << "):\n";
    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < p; ++j) {
            std::cout << "C[" << i + 1 << "][" << j + 1 << "]: " << C[i][
        }
    }
    return 0; // Return success
}
```

TestCases:

S.No	Inputs	Outputs
1	2 2 1 2 3 4 2 2 5 6 7 8	Resulting matrix C (2 x 2): C[1][1]: 19 C[1][2]: 22 C[2][1]: 43 C[2][2]: 50
2	2 3 1 2 3 4 5 6 4 2 7 8 9 10 11 12 13 14	Matrix multiplication is not valid. Number of columns in A must be equal to the number of rows in B.
3	3 2 1 2 3 4 5 6 2 3 7 8 9 10 11 12	Resulting matrix C (3 x 3): C[1][1]: 27 C[1][2]: 30 C[1][3]: 33 C[2][1]: 61 C[2][2]: 68 C[2][3]: 75 C[3][1]: 95 C[3][2]: 106 C[3][3]: 117
4	1 1 5 1 1 3	Resulting matrix C (1 x 1): C[1][1]: 15
5	0 0 0 0	Resulting matrix C (0 x 0):
6	3 3 0 0 0 0 0 0 0 0 0 3 3 1 2 3 4 5 6 7 8 9	Resulting matrix C (3 x 3): C[1][1]: 0 C[1][2]: 0 C[1][3]: 0 C[2][1]: 0 C[2][2]: 0 C[2][3]: 0 C[3][1]: 0 C[3][2]: 0 C[3][3]: 0

White List: inline

Black List:

Question 3:

You are working on a program that requires you to perform type casting operations. Your task is to implement a C++ program that takes user input for an integer x and a double y. Your program should perform the following tasks:

Type Casting:

Convert the integer x to a double and the double y to an integer using appropriate type casting operators.

Calculation:

Calculate the product of the two casted values.

Input Format:

- Prompt the user to enter an integer x.
- Prompt the user to enter a double y.

Output Format:

- Display the integer obtained by casting y to an integer.
- Display the double obtained by casting x to a double with a precision of six decimal places.
- Display the product of the casted values with a precision of six decimal places.

Title for Question 3: Integer to Double Type Casting

Solution:

```
#include <iostream>
#include <iomanip> // For std::fixed and std::setprecision

int main() {
    int x;
    double y;
    // Input: Prompt the user to enter an integer
    std::cin >> x;
    // Input: Prompt the user to enter a double
    std::cin >> y;
    // Cast x to a double
    double castedX = static_cast<double>(x);
    // Cast y to an integer
    int castedY = static_cast<int>(y);
    // Calculate the product of the casted values
    double product = castedX * castedY;
    // Output: Display the casted values and the product
    std::cout << "Casted integer (from y): " << castedY << std::endl;
    std::cout << "Casted double (from x): "  << std::fixed << std::setpre
    std::cout << "Product of casted values: " << std::fixed << std::setpr
    return 0; // Return success
}
```

TestCases:

S.No	Inputs	Outputs
1	5 3.7	Casted integer (from y): 3 Casted double (from x): 5.0000 Product of casted values: 15.0000
2	-8 -2.5	Casted integer (from y): -2 Casted double (from x): -8.0000 Product of casted values: 16.0000
3	0 9.99	Casted integer (from y): 9 Casted double (from x): 0.0000 Product of casted values: 0.0000
4	10000 123456.789	Casted integer (from y): 123456 Casted double (from x): 10000.0000 Product of casted values: 1234560000.0000
5	3 1.41421356	Casted integer (from y): 1 Casted double (from x): 3.0000 Product of casted values: 3.0000
6	0 0.0	Casted integer (from y): 0 Casted double (from x): 0.0000 Product of casted values: 0.0000

White List: static_cast<double>(x),static_cast<int>(y)

Black List:

Question 4:

You are tasked with generating the Fibonacci sequence using both recursion and iteration. Write a Cprogram that performs the following tasks:

- Implement a recursive function named generateFibonacciRecursive that generates and returns the Fibonacci sequence up to the n-th term using recursion. The function should accept the value of n as an argument.
- Implement an iterative function named generateFibonacciIterative that generates and returns the Fibonacci sequence up to the n-th term using regular C++ code. The function should accept the value of n as an argument.

Input Format

- The program prompts the user to enter a positive integer n.
- The user provides a positive integer n as input.

Output Format

- The program displays two lines of output:
- The first line contains the Fibonacci sequence generated using recursion.
- The second line contains the Fibonacci sequence generated using iteration.
- The numbers in the sequences should be separated by spaces.

Title for Question 4: Fibonacci using Recursion and Iteration

Solution:

```
#include <iostream>


// Recursive function to generate Fibonacci sequence
void generateFibonacciRecursive(int n, int a = 0, int b = 1) {
    if (n <= 0) {
        return;
    }
    std::cout << a << " ";
    generateFibonacciRecursive(n - 1, b, a + b);
}


// Iterative function to generate Fibonacci sequence
void generateFibonacciIterative(int n) {
    int a = 0, b = 1;
    for (int i = 0; i < n; ++i) {
        std::cout << a << " ";
        int temp = a;
        a = b;
        b = temp + b;
    }
}


int main() {
```

```
    int n;


    // Prompt the user to enter a positive integer
  //  std::cout << "Enter a positive integer (n): ";
    std::cin >> n;


    // Input validation
    if (n <= 0) {
        std::cerr << "Invalid input. Please enter a positive integer." <<
        return 0; // Return an error code
    }


    // Display the Fibonacci sequence using recursion
    std::cout << "Fibonacci sequence (recursive): ";
    generateFibonacciRecursive(n);


    std::cout << std::endl;


    // Display the Fibonacci sequence using iteration
    std::cout << "Fibonacci sequence (iterative): ";
    generateFibonacciIterative(n);
    std::cout << std::endl;
    return 0; // Return success
}
```

TestCases:

S.No	Inputs	Outputs
1	5	Fibonacci sequence (recursive): 0 1 1 2 3 Fibonacci sequence (iterative): 0 1 1 2 3
2	-3	Invalid input. Please enter a positive integer.
3	10	Fibonacci sequence (recursive): 0 1 1 2 3 5 8 13 21 34 Fibonacci sequence (iterative): 0 1 1 2 3 5 8 13 21 34
4	1	Fibonacci sequence (recursive): 0 Fibonacci sequence (iterative): 0
5	7	Fibonacci sequence (recursive): 0 1 1 2 3 5 8 Fibonacci sequence (iterative): 0 1 1 2 3 5 8
6	14	Fibonacci sequence (recursive): 0 1 1 2 3 5 8 13 21 34 55 89 144 233 Fibonacci sequence (iterative): 0 1 1 2 3 5 8 13 21 34 55 89 144 233

White List:

Black List:

Question 5:

Formulate a program that constructs a pyramid-shaped star pattern using the concept of inline functions. The program should be interactive and request an integer 'n' as input, where 'n' represents the number of rows in the pyramid. Each row of the pyramid should consist of an increasing number of stars, with the stars symmetrically centered within each row. The program

should manage spacing to ensure the stars are properly centered.

Input Format

- The program should prompt the user to enter an integer 'n', representing the number of rows for the pyramid.

Output Format

- The program should display the pyramid-shaped star pattern as output.
- It should print "Star Pattern (Pyramid):" to indicate the start of the pattern.
- The pyramid pattern should be generated with the specified number of rows based on the user's input.
- Each row of the pyramid should contain stars, with proper spacing to ensure center alignment.

Title for Question 5: Pyramid Pattern

Solution:

```
#include <iostream>
// Inline function to print spaces
inline void printSpaces(int spaces) {
    for (int i = 0; i < spaces; i++) {
        std::cout << " ";
    }
}
// Inline function to print stars
inline void printStars(int stars) {
    for (int i = 0; i < stars; i++) {
        std::cout << "*";
    }
    std::cout << std::endl;
}
int main() {
    int n;
    // Input: Ask the user for the number of rows
  //  std::cout << "Enter the number of rows: ";
    std::cin >> n;
    // Output: Print the star pattern (pyramid)
    std::cout << "Star Pattern (Pyramid):\n";
    for (int i = 1; i <= n; i++) {
        printSpaces(n - i);
        printStars(2 * i - 1);
    }
    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	5	Star Pattern (Pyramid): * *** ***** ******* *********
2	4	Star Pattern (Pyramid): * *** ***** *******
3	3	Star Pattern (Pyramid): * *** *****
4	9	Star Pattern (Pyramid): * *** ***** ******* ********* ********** ************ ************** ********** **********
5	5	Star Pattern (Pyramid): * *** ***** ******* *********
6	6	Star Pattern (Pyramid): * *** ***** ******* ********* **********

White List:

Black List:

Question 6:

You are required to implement a C++ program that checks whether a given square matrix qualifies as a "Magic Square." A Magic Square is defined as a square matrix in which the sum of the values in each row, each column, and both main diagonals is the same.

Input Format

- The program expects the following input from the user:
- An integer size (1 <= size <= 10) representing the number of rows and columns in the square matrix.
- size rows, each containing size integers separated by spaces. These values represent the elements of the matrix.

Output Format

- If the matrix is a Magic Square (i.e., the sum of rows, columns, and main diagonals is the same), the program should output: "The matrix is a Magic Square."
- If the matrix is not a Magic Square (i.e., the sums are not equal), the program should output: "The matrix is not a Magic Square."

Title for Question 6: Magic Square Matrix

Solution:

```
#include<iostream>
using namespace std;
// Inline function to allocate memory for a dynamic 2D array
inline int** allocateMatrix(int row) {
    int** arr = new int*[row];
    for (int i = 0; i < row; i++) {
        arr[i] = new int[row];
    }
    return arr;
```

```
}
// Inline function to input values into the matrix
inline void inputMatrix(int** arr, int row) {
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < row; j++) {
            cin >> arr[i][j];
        }
    }
}
// Inline function to calculate the sum of the first row
inline int sumFirstRow(int** arr, int row) {
    int sum = 0;
    for (int i = 0; i < row; i++) {
        sum += arr[0][i];
    }
    return sum;
}
// Inline function to check if a matrix is a Magic Square
inline bool isMagicSquare(int** arr, int row) {
    int sum1 = sumFirstRow(arr, row);
    int sum2 = 0, sum3 = 0;
    // Calculate the sum of the first column and main diagonal
    for (int i = 0; i < row; i++) {
        sum2 += arr[i][0];
        sum3 += arr[i][i];
    }
    return (sum1 == sum2 && sum2 == sum3);
}
int main() {
    int row;
    cin >> row;
    // Allocate memory for the dynamic 2D array
    int** arr = allocateMatrix(row);
    // Input values into the matrix
    inputMatrix(arr, row);
    // Check if the matrix is a Magic Square using the inline function
    if (isMagicSquare(arr, row)) {
        cout << "Magic Square" << endl;
    } else {
        cout << "Not a Magic Square" << endl;
    }
    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	3 2 7 6 9 5 1 4 3 8	Magic Square
2	4 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	Not a Magic Square
3	3 2 7 6 9 4 1 4 3 7	Not a Magic Square
4	5 17 24 1 8 15 23 5 7 14 16 4 6 13 20 22 10 12 19 21 3 11 18 25 2 9	Magic Square
5	1 1	Magic Square
6	2 1 2 1 2	Not a Magic Square

White List:

Black List: