Question 1:

Write a program that calculates and displays the squares of multiple numbers entered by the user. The program should keep taking input until the user decides to exit and . The user should be able to enter as many numbers as they want, and the program should display the square of each number. Additionally, the program should keep track of the total number of squares calculated and display it when the user chooses to exit.

Function Header:

```
double square(double number)
```

Input Format:

- The program should prompt the user to enter an integer 'number'.

Output Format:

- Display the result in the format: "The square of <number> is : <n>"

Title for Question 1: Square of Number

Solution:

```
#include <iostream>
double square(double number)
{
    return (number * number);
}
int main() {
    int number;
    double n;
    std::cin >> number;
    n = square(number);
    std::cout << "The square of " << number << " is : " << n << std::endl
    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	5	The square of 5 is : 25

S.No	Inputs	Outputs
2	-3	The square of -3 is : 9
3	123	The square of 123 is : 15129
4	34	The square of 34 is : 1156
5	9	The square of 9 is : 81
6	10	The square of 10 is : 100

White List: double square(double number)

Black List:

Question 2:

Create a program that repeatedly calculates the sum of integers from 1 to a specified positive integer n, taking input from the user until the user decides to exit. The program should display the sum for each input, and the user can continue or exit after each calculation. If the user enters a non-positive integer, the program should display an error message and ask for a valid input. The program should also handle non-integer inputs gracefully.

Function Header:

```
int add(int n)
```

?????Input Format:

- The program expects the user to enter a single positive integer n (greater than or equal to 0).
- If the input n is negative, the program will display "Invalid Input" and exit.

Output Format:

- The program displays the message: "The sum of the series is : <sum>".
- If the input n is negative, the program will display "Invalid Input.".

Title for Question 2: Sum of the series

Solution:

```
#include <iostream>
using namespace std;
int add(int);
int main() {
    int n, sum = 0;
    cin >> n;
    if(n<0){
        cout<<"Invalid Input";
        exit(0) ;
```

```
    }else{
    sum = add(n);
}
    cout << "The sum of the series is : " << sum << "\n\n";
    return 0;
}
int add(int n)
{
    int plus;
    for(int i=1;i<=n;i++){
        plus  +=i;
    }
    return plus;
}
```

TestCases:

S.No	Inputs	Outputs
1	5	The sum of the series is : 15
2	-16	Invalid Input
3	0	The sum of the series is : 0
4	24	The sum of the series is : 300
5	-87	Invalid Input
6	-90	Invalid Input

White List:

Black List:

Question 3:

Modify the given code to allow the user to choose between calculating the area of a rectangle or a circle based on their input. The program should dynamically adapt to the user's choice and then request the necessary input accordingly.

Input Format:

For calculating the area of a rectangle:

- The program prompts the user to enter the length of the rectangle.
- The program prompts the user to enter the width of the rectangle.
- The user provides these values as floating-point numbers.

For calculating the area of a circle:

- The program prompts the user to enter the radius of the circle.
- The user provides this value as a floating-point number.

Output Format:

For calculating the area of a rectangle:

- The program displays the area of the rectangle in the format: "Area of the rectangle: <area>".

For calculating the area of a circle:

- The program displays the area of the circle in the format: "Area of the circle: <area>".

Title for Question 3: Calculating Area

Solution:

```
#include <iostream>
#include <cmath>
const double PI = 3.14159265359;
double calculateArea(double length, double width) {
    return length * width;
}
double calculateArea(double radius) {
    return PI * pow(radius, 2);
}
int main() {
    double rectangleLength, rectangleWidth;
    double circleRadius;


    std::cin >> rectangleLength;
    std::cin >> rectangleWidth;
    double rectangleArea = calculateArea(rectangleLength, rectangleWidth);
    std::cout << "Area of the rectangle: " << rectangleArea << std::endl;
    std::cin >> circleRadius;
    double circleArea = calculateArea(circleRadius);
    std::cout << "Area of the circle: " << circleArea << std::endl;
    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	10 15 20	Area of the rectangle: 150 Area of the circle: 1256.64
2	15.0 3.78 24.5	Area of the rectangle: 56.7 Area of the circle: 1885.74
3	8.7 2.5 18.75	Area of the rectangle: 21.75 Area of the circle: 1104.47
4	0.0 0.0 0.0	Area of the rectangle: 0 Area of the circle: 0
5	10.25 4.5 8.0	Area of the rectangle: 46.125 Area of the circle: 201.062
6	21.2 30.6 27.0	Area of the rectangle: 648.72 Area of the circle: 2290.22

White List: const double PI = 3.14159265359;

Black List:

Question 4:

Create a program that takes two integer inputs, start and end, representing a range of even numbers. The program should find all numbers within this range that can be expressed as the sum of two prime numbers (e.g., 10 = 3 + 7) using the checkPrime function provided in the code.

Note:

Print only one combination of the sum as shown in the sample output. Use function checkPrime .

Input Format

- Starting number as an integer
- Ending number as an integer

Output Format

- Prime number representation of all even numbers between the given range as shown in the sample output

Title for Question 4: Range of Even Number

Solution:

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

int checkPrime(int n)
{
    int flag=0;
    for(int i = 2; i <= n/2; ++i)
    {
        if(n%i == 0)
        {
            flag = 1;
            break;
        }
    }
    if(flag==0)
        return 1;
    else
        return 0;
}
```

```
int main()
{
    int n,flag =0,start,end,k=0;
    cin>>start>>end;

    int m=(end-start)/2;
    int array[m][3];
    for(n=start;n<=end;n+=2)
 {
     for(int i=2;i<n/2;i++){
        flag=checkPrime(n-i);
        if(flag==1){

            array[k][0]=n;
            array[k][1]=n-i;
            array[k][2]=i;
            k++;


            break;
        }
    }
}


for(int i=0;i<k;i++)
    cout<<array[i][0]<<" = "<<array[i][1]<<" + "<<array[i][2]<<endl;
    //printf("%d = %d + %d\n",array[i][0],array[i][1],array[i][2]);
}
```

TestCases:

S.No	Inputs	Outputs
1	20 30	20 = 17 + 3 22 = 19 + 3 24 = 19 + 5 26 = 23 + 3 28 = 23 + 5 30 = 23 + 7
2	10 30	10 = 7 + 3 12 = 7 + 5 14 = 11 + 3 16 = 13 + 3 18 = 13 + 5 20 = 17 + 3 22 = 19 + 3 24 = 19 + 5 26 = 23 + 3 28 = 23 + 5 30 = 23 + 7
3	10 40	10 = 7 + 3 12 = 7 + 5 14 = 11 + 3 16 = 13 + 3 18 = 13 + 5 20 = 17 + 3 22 = 19 + 3 24 = 19 + 5 26 = 23 + 3 28 = 23 + 5 30 = 23 + 7 32 = 29 + 3 34 = 31 + 3 36 = 31 + 5 38 = 31 + 7 40 = 37 + 3
4	30 50	30 = 23 + 7 32 = 29 + 3 34 = 31 + 3 36 = 31 + 5 38 = 31 + 7 40 = 37 + 3 42 = 37 + 5 44 = 41 + 3 46 = 43 + 3 48 = 43 + 5 50 = 47 + 3
5	20 30	20 = 17 + 3 22 = 19 + 3 24 = 19 + 5 26 = 23 + 3 28 = 23 + 5 30 = 23 + 7
6	12 18	12 = 7 + 5 14 = 11 + 3 16 = 13 + 3 18 = 13 + 5

White List: int checkPrime(int n)

Black List:

Question 5:

Write a program that takes two strings as input from the user and uses the checkAnagram function to determine if they are anagrams. Additionally, the program should be able to handle various edge

cases and provide appropriate error messages or results.

Input Format:

- The program expects two lines of input.
- The first line should contain the first string (str1).
- The second line should contain the second string (str2).

Output Format:

The program will print one of the following messages:

- If the two strings are anagrams, it will print: "str1 and str2 are Anagrams."
- If the two strings are not anagrams, it will print: "str1 and str2 are not Anagrams."

Constraints:

- The input strings (str1 and str2) may contain alphabetic characters (both uppercase and lowercase) and spaces.
- The maximum length of each input string is 256 characters.

Title for Question 5: Check a String is Anagram

Solution:

```
#include <iostream>
#include <string>
bool checkAnagram(const std::string& str1, const std::string& str2) {
    if (str1.length() != str2.length()) {
        return false;
    }
    int str1ChrCtr[256] = {0};
    int str2ChrCtr[256] = {0};
    for (int i = 0; i < str1.length(); i++) {
        char c = str1[i];
        str1ChrCtr[static_cast<int>(c)]++;
    }
    for (int i = 0; i < str2.length(); i++) {
        char c = str2[i];
        str2ChrCtr[static_cast<int>(c)]++;
    }
    for (int i = 0; i < 256; i++) {
        if (str1ChrCtr[i] != str2ChrCtr[i]) {
            return false;
        }
    }
    return true;
}
int main() {
```

```
    std::string str1, str2;
    std::getline(std::cin, str1);
    std::getline(std::cin, str2);
    if (checkAnagram(str1, str2)) {
        std::cout << " " << str1 << " and " << str2 << " are Anagrams." <
    } else {
        std::cout << " " << str1 << " and " << str2 << " are not Anagrams
    }
    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	listen silent	listen and silent are Anagrams.
2	apple banana	apple and banana are not Anagrams.
3	Hello heLlo	Hello and heLlo are not Anagrams.
4	"" ""	"" and "" are Anagrams.
5	"" "hello"	"" and "hello" are not Anagrams.
6	hello123 123world	hello123 and 123world are not Anagrams.

White List:

Black List:

Question 6:

Write a code to make it work as expected for swapping two numbers using a function. However, you are not allowed to use pointers in the swap function. Instead, you should come up with an alternative approach to achieve the same result.

Function Header

```
void swap(int p, int q)
```

Input Format:

- The program expects two integers separated by spaces or newlines.
- The user enters the first integer.
- The user enters the second integer.

Output Format:

- The program directly reads the input values without displaying prompts.
- After reading both integers, it displays the original values of n1 and n2 before swapping.
- Then, it displays the values of n1 and n2 after swapping.

Title for Question 6: Swap an Integer

Solution:

```
#include <iostream>
void swap(int p, int q );
int main() {
    int n1, n2;
    //std::cout << "Input 1st number : ";
    std::cin >> n1;
   // std::cout << "Input 2nd number : ";
    std::cin >> n2;
    std::cout << "Before swapping: n1 = " << n1 << ", n2 = " << n2;
    // Pass the address of both variables to the function.
    swap(n1, n2);

    return 0;
}
void swap(int p, int q) {
    int tmp;
    tmp = p;   // tmp stores the value of n1
    p = q;    // p stores the value of q, which is the value of n2
    q = tmp;   // q stores the value of tmp, which is the value of n1
      std::cout << "\nAfter swapping: n1 = " << p << ", n2 = " << q << "\
}
```

TestCases:

S.No	Inputs	Outputs
1	10 20	Before swapping: n1 = 10, n2 = 20 After swapping: n1 = 20, n2 = 10
2	-29 9	Before swapping: n1 = -29, n2 = 9 After swapping: n1 = 9, n2 = -29
3	0 0	Before swapping: n1 = 0, n2 = 0 After swapping: n1 = 0, n2 = 0
4	123456 654321	Before swapping: n1 = 123456, n2 = 654321 After swapping: n1 = 654321, n2 = 123456
5	0 -10	Before swapping: n1 = 0, n2 = -10 After swapping: n1 = -10, n2 = 0
6	123 -456	Before swapping: n1 = 123, n2 = -456 After swapping: n1 = -456, n2 = 123

White List: void swap(int p, int q)

Black List:
