Question 1:

Imagine you are developing a software module for a scientific calculator. Users will input a base (a) and an exponent (b), and your module should calculate and display the result of raising the base to the power of the exponent.

Input Format:

- Prompt the user to enter the base (a) as an integer.
- Prompt the user to enter the exponent (b) as an integer.

Output Format:

- Display the result of raising the base to the power of the exponent in the format: "{a} raised to the power of {b} is: {result}".

Title for Question 1: Exponential Function

Solution:

```
#include <iostream>

long long power(int a, int b) {
    if (b == 0) {
        return 1;
    }

    long long halfPower = power(a, b / 2);

    if (b % 2 == 0) {
        // If b is even
        return halfPower * halfPower;
    } else {
        // If b is odd
        return halfPower * halfPower * a;
    }
}

int main() {
    int a, b;
  //  std::cout << "Enter the base (a): ";
    std::cin >> a;
//    std::cout << "Enter the exponent (b): ";
    std::cin >> b;

    long long result = power(a, b);

    std::cout << a << " raised to the power of " << b << " is: " << resul

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	2 3	2 raised to the power of 3 is: 8
2	5 0	5 raised to the power of 0 is: 1
3	3 4	3 raised to the power of 4 is: 81
4	1 10	1 raised to the power of 10 is: 1
5	4 5	4 raised to the power of 5 is: 1024
6	7 2	7 raised to the power of 2 is: 49

White List:

Black List:

Question 2:

Implement the Tower of Hanoi problem using recursion. The program should take an input, numDisks, representing the number of disks in the Tower of Hanoi. Implement the towerOfHanoi function to print the sequence of moves required to solve the Tower of Hanoi problem.

Input Format:

- An integer numDisks representing the number of disks in the Tower of Hanoi.

Output Format:

- Print each move in the format "Move disk x from source to destination" where x is the disk number, and source and destination are the source and destination pegs (A, B, or C).

Title for Question 2: Tower of Hanoi

Solution:

```
#include<iostream>

using namespace std;

void hanoi(int n,char s, char a,char d){
    if (n==1){
        cout<<"Move disk 1 from "<<s <<" to "<<d<<endl;
        return;
    }

    hanoi(n - 1,s,d,a);
    cout<<"Move disk "<<n<<" from "<<s<<" to "<<d<<endl;

    hanoi(n - 1,a,s,d);
}
```

```
int main(){

    int disk;

    cin>>disk;

    hanoi(disk,'A','B','C');

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	1	Move disk 1 from A to C
2	3	Move disk 1 from A to C Move disk 2 from A to B Move disk 1 from C to B Move disk 3 from A to C Move disk 1 from B to A Move disk 2 from B to C Move disk 1 from A to C
3	5	Move disk 1 from A to C Move disk 2 from A to B Move disk 1 from C to B Move disk 3 from A to C Move disk 1 from B to A Move disk 2 from B to C Move disk 1 from A to C Move disk 4 from A to B Move disk 1 from C to B Move disk 2 from C to A Move disk 1 from B to A Move disk 3 from C to B Move disk 1 from A to C Move disk 2 from A to B Move disk 1 from C to B Move disk 5 from A to C Move disk 1 from B to A Move disk 2 from B to C Move disk 1 from A to C Move disk 3 from B to A Move disk 1 from C to B Move disk 2 from C to A Move disk 1 from B to A Move disk 4 from B to C Move disk 1 from A to C Move disk 2 from A to B Move disk 1 from C to B Move disk 3 from A to C Move disk 1 from B to A Move disk 2 from B to C Move disk 1 from A to C
4	2	Move disk 1 from A to B Move disk 2 from A to C Move disk 1 from B to C

S.No	Inputs	Outputs
5	7	Move disk 1 from A to C Move disk 2 from A to B Move disk 1 from C to B Move disk 3 from A to C Move disk 1 from B to A Move disk 2 from B to C Move disk 1 from A to C Move disk 4 from A to B Move disk 1 from C to B Move disk 2 from C to A Move disk 1 from B to A Move disk 3 from C to B Move disk 1 from A to C Move disk 2 from A to B Move disk 1 from C to B Move disk 5 from A to C Move disk 1 from B to A Move disk 2 from B to C Move disk 1 from A to C Move disk 3 from B to A Move disk 1 from C to B Move disk 2 from C to A Move disk 1 from B to A Move disk 4 from B to C Move disk 1 from A to C Move disk 2 from A to B Move disk 1 from C to B Move disk 3 from A to C Move disk 1 from B to A Move disk 2 from B to C Move disk 1 from A to C Move disk 6 from A to B Move disk 1 from C to B Move disk 2 from C to A Move disk 1 from B to A Move disk 3 from C to B Move disk 1 from A to C Move disk 2 from A to B Move disk 1 from C to B Move disk 4 from C to A Move disk 1 from B to A Move disk 2 from B to C Move disk 1 from A to C Move disk 3 from B to A Move disk 1 from C to B Move disk 2 from C to A Move disk 1 from B to A Move disk 5 from C to B Move disk 1 from A to C Move disk 2 from A to B Move disk 1 from C to B Move disk 3 from A to C Move disk 1 from B to A Move disk 2 from B to C Move disk 1 from A to C Move disk 4 from A to B Move disk 1 from C to B Move disk 2 from C to A Move disk 1 from B to A Move disk 3 from C to B Move disk 1 from A to C Move disk 2 from A to B Move disk 1 from C to B Move disk 7 from A to C Move disk 1 from B to A Move disk 2 from B to C Move disk 1 from A to C Move disk 3 from B to A Move disk 1 from C to B Move disk 2 from C to A Move disk 1 from B to A Move disk 4 from B to C Move disk 1 from A to C Move disk 2 from A to B Move disk 1 from C to B Move disk 3 from A to C Move disk 1 from B to A Move disk 2 from B to C Move disk 1 from A to C Move disk 5 from B to A Move disk 1 from C to B Move disk 2 from C to A Move disk 1 from B to A Move disk 3 from C to B Move disk 1 from A to C Move disk 2 from A to B Move disk 1 from C to B Move disk 4 from C to A Move disk 1 from B to A Move disk 2 from B to C Move disk 1 from A to C Move disk 3 from B to A Move disk 1 from C to B Move disk 2 from C to A Move disk 1 from B to A Move disk 6 from B to C Move disk 1 from A to C Move disk 2 from A to B Move disk 1 from C to B Move disk 3 from A to C Move disk 1 from B to A Move disk 2 from B to C Move disk 1 from A to C Move disk 4 from A to B Move disk 1 from C to B Move disk 2 from C to A Move disk 1 from B to A Move disk 3 from C to B Move disk 1 from A to C Move disk 2 from A to B Move disk 1 from C to B Move disk 5 from A to C Move disk 1 from B to A Move disk 2 from B to C Move disk 1 from A to C Move disk 3 from B to A Move disk 1 from C to B Move disk 2 from C to A Move disk 1 from B to A Move disk 4 from B to C Move disk 1 from A to C Move disk 2 from A to B Move disk 1 from C to B Move disk 3 from A to C Move disk 1 from B to A Move disk 2 from B to C Move disk 1 from A to C

S.No	Inputs	Outputs
6	6	Move disk 1 from A to B Move disk 2 from A to C Move disk 1 from B to C Move disk 3 from A to B Move disk 1 from C to A Move disk 2 from C to B Move disk 1 from A to B Move disk 4 from A to C Move disk 1 from B to C Move disk 2 from B to A Move disk 1 from C to A Move disk 3 from B to C Move disk 1 from A to B Move disk 2 from A to C Move disk 1 from B to C Move disk 5 from A to B Move disk 1 from C to A Move disk 2 from C to B Move disk 1 from A to B Move disk 3 from C to A Move disk 1 from B to C Move disk 2 from B to A Move disk 1 from C to A Move disk 4 from C to B Move disk 1 from A to B Move disk 2 from A to C Move disk 1 from B to C Move disk 3 from A to B Move disk 1 from C to A Move disk 2 from C to B Move disk 1 from A to B Move disk 6 from A to C Move disk 1 from B to C Move disk 2 from B to A Move disk 1 from C to A Move disk 3 from B to C Move disk 1 from A to B Move disk 2 from A to C Move disk 1 from B to C Move disk 4 from B to A Move disk 1 from C to A Move disk 2 from C to B Move disk 1 from A to B Move disk 3 from C to A Move disk 1 from B to C Move disk 2 from B to A Move disk 1 from C to A Move disk 5 from B to C Move disk 1 from A to B Move disk 2 from A to C Move disk 1 from B to C Move disk 3 from A to B Move disk 1 from C to A Move disk 2 from C to B Move disk 1 from A to B Move disk 4 from A to C Move disk 1 from B to C Move disk 2 from B to A Move disk 1 from C to A Move disk 3 from B to C Move disk 1 from A to B Move disk 2 from A to C Move disk 1 from B to C

White List:

Black List:

Question 3:

Design a program that calculates the Ackermann function A(m, n) using a recursive approach. The Ackermann function is defined as follows:

n + 1 & \text{if } m = 0 \\

A(m - 1, 1) & \text{if } m > 0 \text{ and } n = 0 \\

A(m - 1, A(m, n - 1)) & \text{if } m > 0 \text{ and } n > 0 \\

\end{cases}

The program prompts the user to enter values for m and n and then computes and displays the result of A(m, n).

Input Format:

- The program takes two non-negative integers, m and n, as input.
- The input is provided through standard input (stdin).

Output Format:

- The program outputs the result of the Ackermann function A(m, n).
- If the input is invalid (negative integers), it outputs an error message like “Invalid input. Please enter non-negative integers for m and n.”.

Title for Question 3: Ackermann Function

Solution:

```
#include <iostream>

int Ackermann(int m, int n) {
    if (m == 0) {
        return n + 1;
    } else if (m > 0 && n == 0) {
        return Ackermann(m - 1, 1);
    } else if (m > 0 && n > 0) {
        return Ackermann(m - 1, Ackermann(m, n - 1));
    } else {
        // Invalid input
        return -1;
    }
}

int main() {
    int m, n;
   // std::cout << "Enter values for m and n: ";
    std::cin >> m >> n;

    int result = Ackermann(m, n);

    if (result != -1) {
        std::cout << "A(" << m << ", " << n << ") = " << result << std::e
    } else {
        std::cout << "Invalid input. Please enter non-negative integers f
    }

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	1 1	A(1, 1) = 3
2	-2 2	Invalid input. Please enter non-negative integers for m and n.
3	2 3	A(2, 3) = 9
4	3 2	A(3, 2) = 29
5	0 5	A(0, 5) = 6
6	-4 0	Invalid input. Please enter non-negative integers for m and n.

White List:

Black List:

Question 4:

You are tasked with optimizing a Fibonacci calculation module for a software application. The current implementation includes both a regular recursive function and a memoized version. Your goal is to understand the impact of memoization on performance.

Input Format:

- Prompt the user to enter an integer value 'n' representing the position in the Fibonacci sequence.

Output Format:

- Display the result of the Fibonacci calculation with memoization in the format: "Fibonacci(n) with memoization: result".

Title for Question 4: Recursive Function Optimization

Solution:

```
#include <iostream>
#include <cstring> // For memset
using namespace std;

const int MAX = 1000; // Maximum size of Fibonacci numbers we are willing
int memo[MAX];

int fibonacciMemoized(int n) {
    if (n <= 1) {
        return n;
    }

    // Check if the result is already computed
    if (memo[n] != -1) {
        return memo[n];
    }

    // Compute and remember the result
    memo[n] = fibonacciMemoized(n - 1) + fibonacciMemoized(n - 2);
    return memo[n];
}

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

    // Initialize memoization array with -1 to indicate uncomputed values
    memset(memo, -1, sizeof(memo));

    cout << "Fibonacci(" << n << ") with memoization: " << fibonacciMemoi
    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	5	Fibonacci(5) with memoization: 5
2	-9	Fibonacci(-9) with memoization: -9
3	4	Fibonacci(4) with memoization: 3
4	-2	Fibonacci(-2) with memoization: -2
5	6	Fibonacci(6) with memoization: 8
6	3	Fibonacci(3) with memoization: 2

White List:

Black List:

Question 5:

Create a program that calculates the sum of the digits of a given non-negative integer using function named sumOfDigits(). The input will be a single non-negative integer, and the output should be the sum of its digits. You should handle any incorrect inputs gracefully, informing the user of the invalid input.

Input Format

- The input contains a single line with a non-negative integer N (0 ? N ? 10^9).

Output Format

- If the input is valid (a non-negative integer), print the sum of the digits of N, formatted as: "The sum of digits of N is X", where N is the input number and X is the calculated sum of its digits.
- If the input is invalid (a negative integer), print: "Invalid input. Please enter a non-negative integer."

Title for Question 5: Sum of digits

Solution:

```
#include <iostream>

// Function declaration
```

```
int sumOfDigits(int n);

int main() {
    int number;
   // std::cout << "Enter a number: ";
    std::cin >> number;

    // Check if the input is successful and the number is non-negative
    if (number < 0) {
        std::cout << "Invalid input. Please enter a non-negative integer.
        return 1;  // Return with error code
    }

    std::cout << "The sum of digits of " << number << " is " << sumOfDigi
    return 0;
}

// Recursive function to calculate the sum of digits of a number
int sumOfDigits(int n) {
    // Base case: When n is reduced to 0
    if (n == 0) return 0;

    // Step case: Add the last digit to the sum of the remaining digits
    return (n % 10) + sumOfDigits(n / 10);
}
```

TestCases:

S.No	Inputs	Outputs
1	2311	The sum of digits of 2311 is 7
2	-27	Invalid input. Please enter a non-negative integer.
3	413	The sum of digits of 413 is 8
4	11	The sum of digits of 11 is 2
5	271	The sum of digits of 271 is 10
6	-9	Invalid input. Please enter a non-negative integer.

White List: sumOfDigits

Black List:

Question 6:

Implement a program that checks whether there exists a subset of the given set whose sum is equal to the target sum. The program should take input for the size of the set, the elements of the set, and the target sum. It should then output whether a subset with the target sum exists or not.

Input Format:

- The first line contains an integer 'size' (1 <= size <= 20), representing the size of the set.
- The second line contains 'size' space-separated integers, representing the elements of the set.

- The third line contains an integer 'targetSum' (1 <= targetSum <= 1000), representing the target sum.

Output Format:

- If there exists a subset with the target sum, output: "Subset with the target sum exists."
- If no subset with the target sum exists, output: "No subset with the target sum exists."

Title for Question 6: Subset Sum

Solution:

```
#include <iostream>
#include <vector>

bool subsetSumExists(const std::vector<int>& nums, int targetSum, int cur
    if (currentSum == targetSum) {
        return true;
    }

    if (currentIndex == nums.size()) {
        return false;
    }

    bool includeCurrent = subsetSumExists(nums, targetSum, currentIndex +

    bool excludeCurrent = subsetSumExists(nums, targetSum, currentIndex +

    return includeCurrent || excludeCurrent;
}

int main() {
    std::vector<int> nums;

    int size;
    std::cin >> size;

    for (int i = 0; i < size; ++i) {
        int num;
        std::cin >> num;
        nums.push_back(num);
    }

    int targetSum;
    std::cin >> targetSum;

    if (subsetSumExists(nums, targetSum)) {
        std::cout << "Subset with the target sum exists." << std::endl;
    } else {
        std::cout << "No subset with the target sum exists." << std::endl
    }

    return 0;
}
```

TestCases:

S.No	Inputs	Outputs
1	5 1 2 3 4 5 10	Subset with the target sum exists.
2	3 5 7 9 11	No subset with the target sum exists.
3	4 10 20 30 40 60	Subset with the target sum exists.
4	2 3 8 12	No subset with the target sum exists.
5	6 1 2 3 4 5 6 20	Subset with the target sum exists.
6	4 -1 2 -3 4 1	Subset with the target sum exists.

White List:

Black List:
