

## Question01.cpp

```
1 // DSA LAB 7
2 // <---Q1--->
3 // Write a program and recurrence relation to find the Fibonacci series of n where n>2.
4
5 #include <iostream>
6 using namespace std;
7
8 // Function to find the nth Fibonacci number using recursion
9 int fibonacciRecursive(int n) {
10     if (n <= 1) {
11         return n;
12     } else {
13         return fibonacciRecursive(n - 1) + fibonacciRecursive(n - 2);
14     }
15 };
16
17 int main() {
18     int n;
19
20     // Input the value of n
21     cout << "Enter the value of n (n > 2): ";
22     cin >> n;
23
24     // Check if n is greater than 2
25     if (n <= 2) {
26         cout << "Invalid input. Please enter n > 2." << endl;
27         return 1; // Return an error code
28     }
29
30     // Using recursion to find the nth Fibonacci number
31     cout << "Fibonacci series using recursion:" << endl;
32     for (int i = 0; i < n; ++i) {
33         cout << fibonacciRecursive(i) << " ";
34     }
35     cout << endl;
36
37     return 0;
38 }
39
```

## Question02.cpp

```
1 // DSA LAB 7
2 // <---Q2--->
3
4 // Write a program and recurrence relation to find the Factorial of n where n>2.
5
6 #include <iostream>
7 using namespace std;
8
9 // Function to find the factorial of a number using recursion
10 unsigned long long factorialRecursive(int n)
11 {
12     if (n <= 1)
13     {
14         return 1;
15     }
16     else
17     {
18         return n * factorialRecursive(n - 1);
19     }
20 }
21
22 int main()
23 {
24     int n;
25
26     // Input the value of n
27     cout << "Enter the value of n (n > 2): ";
28     cin >> n;
29
30     // Check if n is greater than 2
31     if (n <= 2)
32     {
33         cout << "Invalid input. Please enter n > 2." << endl;
34         return 1; // Return an error code
35     }
36
37     // Using recursion to find the factorial of n
38     cout << "Factorial using recursion: " << factorialRecursive(n) << endl;
39     return 0;
40 }
41
```

## Question03.cpp

```
1 // DSA LAB 7
2 // <---Q3--->
3
4 // Write a recursive function which will take input from the user until a special character
5 // (also selected by the user) is not entered. Then print all the input in reverse.
6 // Sample Input:
7 // Enter Special Character: !
8 // Enter Character: A
9 // Enter Character: B
10 // Enter Character: C
11 // Enter Character: !
12 // Sample Output: C B A
13
14
15 #include <iostream>
16 #include <string>
17
18 using namespace std;
19
20 // Recursive function to read input until a special character is encountered
21 void readInputAndReverse(char specialCharacter)
22 {
23     char ch;
24     cout << "Enter Character: ";
25     cin >> ch;
26
27     // Check if the entered character is the special character
28     if (ch == specialCharacter)
29     {
30         // Base case: Stop recursion
31         return;
32     }
33
34     // Recursive call for the next character
35     readInputAndReverse(specialCharacter);
36
37     // Print the entered character after the recursive call (post-order)
38     cout << ch << " ";
39 }
40
41 int main()
42 {
43     char specialCharacter;
44
45     // Input the special character
46     cout << "Enter Special Character: ";
47     cin >> specialCharacter;
48     cout << endl;
49
50     // Read input until the special character is encountered and print in reverse
51     readInputAndReverse(specialCharacter);
52     cout << "Sample Output ";
53     cout << endl;
```

```
54  
55     return 0;  
56 }  
57
```

## Question04.cpp

```
1 // DSA LAB 7
2 // <---Q4--->
3
4 // Write a recursive function which will raise a number (double) to a non-negative integer
5 // power n. The function receives the double value and integer as arguments.
6
7 #include <iostream>
8
9 using namespace std;
10
11 // Recursive function to calculate the power of a double value
12 double power(double base, int exponent) {
13     // Base case: Any number raised to the power of 0 is 1
14     if (exponent == 0) {
15         return 1.0;
16     }
17
18     // Recursive case: Multiply the base by the result of the recursive call
19     // with a reduced exponent
20     return base * power(base, exponent - 1);
21 }
22
23 int main() {
24     double base;
25     int exponent;
26
27     // Input the base and exponent
28     cout << "Enter the base (double): ";
29     cin >> base;
30
31     cout << "Enter the exponent (non-negative integer): ";
32     cin >> exponent;
33
34     // Check if the exponent is non-negative
35     if (exponent < 0) {
36         cout << "Invalid exponent. Please enter a non-negative integer." << endl;
37         return 1; // Return an error code
38     }
39
40     // Calculate and print the result
41     double result = power(base, exponent);
42     cout << base << " raised to the power " << exponent << " is: " << result << endl;
43
44     return 0;
45 }
46
```

## Question05.cpp

```
1 // DSA LAB 7
2 // <---Q5--->
3
4 // Write a recursive method that for a positive integer n prints odd numbers
5
6 // a. between 1 and n
7 // b. between n and 1
8
9
10 #include <iostream>
11 using namespace std;
12 void printOddNumbersUpToN(int n) {
13     if (n >= 1) {
14         printOddNumbersUpToN(n - 1);
15         if (n % 2 != 0) {
16             cout << n << " ";
17         }
18     }
19 }
20 void printOddNumbersDownTo1(int n) {
21     if (n >= 1) {
22         if (n % 2 != 0) {
23             cout << n << " ";
24         }
25         printOddNumbersDownTo1(n - 1);
26     }
27 }
28 int main() {
29     int n;
30     do {
31         cout << "Enter a positive integer (n): ";
32         cin >> n;
33     } while (n <= 0);
34     cout << "a. Odd numbers between 1 and " << n << ": ";
35     printOddNumbersUpToN(n);
36     cout << "\nb. Odd numbers between " << n << " and 1: ";
37     printOddNumbersDownTo1(n);
38     cout << endl;
39     return 0;
40 }
41
```