First C++ Project

Mohammadreza Aliyari,

SBU ID: 115908077

Q.1:

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

int main() {
    int n;

    // Prompting user for input
    cout << "Enter a number: ";
    cin >> n; // Taking input from user

    // Switch-case statement to check the value of 'n'
    switch (n) {
        case -1:
            cout << "negative one" << endl; // Display if 'n' is -1
            break;
        case 0:
            cout << "zero" << endl; // Display if 'n' is 0
            break;
        case 1:
            cout << "positive one" << endl; // Display if 'n' is 1
            break;
        default:
            cout << "other value" << endl; // Display for any other 'n'
    }

    return 0; // Return 0 indicating successful execution
}
```

This code prompts the user for input, reads an integer, and then uses a switch-case statement to print specific messages based on the value of the entered number. Each case handles a different value of 'n', and the default case catches any value that doesn't match the specified cases.

Q.2:

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

// Function to print the elements of a vector
void print(std::vector<int>& v) {
    std::cout << "The vector elements are : ";

    // Loop through the vector and print each element
    for (int i = 0; i < v.size(); i++)
        std::cout << v.at(i) << ' ';

    std::cout << std::endl; // Adding a newline for clarity
}

int main() {
    std::vector<int> v = {2, 4, 3, 5, 6}; // Creating a vector with some elements
    print(v); // Calling the print function to display the vector elements
    return 0;
}
```

This code defines a function print that takes a reference to a vector of integers and prints its elements. In the main function, it creates a vector v with some elements and calls the print function to display the elements of the vector. The function print iterates through the vector and prints each element separated by spaces, followed by a newline for clarity.

Q3:

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

int main() {
    int t1 = 0, t2 = 1, nextTerm = 0;
    int limit = 4000000; // The limit to not exceed in the Fibonacci sequence

    cout << "Fibonacci Series: ";

    while (nextTerm <= limit) { // Loop until the next term exceeds the limit
        if (nextTerm != 0) {
            cout << nextTerm << ", "; // Print the next term if it's not zero
        }

        nextTerm = t1 + t2; // Calculate the next term in the sequence
        t1 = t2; // Shift the values for the next iteration
        t2 = nextTerm; // Shift the values for the next iteration
    }

    cout << "..." << endl; // Printing ellipsis to indicate the sequence exceeds the limit
    return 0;
}
```

This code utilizes a while loop to generate the Fibonacci sequence, printing each term that does not exceed the limit of 4,000,000. Here's how it works:

1- It initializes variables t1, t2, and ‘nextTerm’ to handle the Fibonacci sequence.
2- The while loop continues until ‘nextTerm’ exceeds the limit of 4,000,000.
3- Inside the loop, it calculates the next term by adding t1 and t2, updates the variables accordingly, and prints the ‘nextTerm’ if it's within the limit.
4- Finally, it prints an ellipsis to indicate that the sequence exceeds the limit.

Q.4:

Q4a:

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

bool isprime(int n) {
    bool result = true; // Assume the number is prime initially

    if (n <= 1) { // Check if the number is less than or equal to 1
        result = false; // Numbers less than or equal to 1 are not prime
    } else {
        // Check divisibility from 2 up to n-1
        for (int i = 2; i < n; i++) {
            if (n % i == 0) { // If divisible by any number other than 1 and itself
                result = false; // Set result to false as it's not prime
                break; // Exit the loop, no need to check further
            }
        }
    }

    // Print whether the number is prime or not (Note: This part shouldn't be in the function)
    if (result == true) {
        cout << "Number is prime ";
    } else {
        cout << "Number is not prime ";
    }

    return result; // Return the boolean result indicating if the number is prime or not
}

void test_isprime() {
    // Testing the isprime function with different numbers
    std::cout << "isprime(2) = " << isprime(2) << '\n'; // Expected: 1 (true)
    std::cout << "isprime(10) = " << isprime(10) << '\n'; // Expected: 0 (false)
    std::cout << "isprime(17) = " << isprime(17) << '\n'; // Expected: 1 (true)
}

int main() {
    test_isprime(); // Calling the test function to run the test cases
    return 0;
}
```

Regarding this code: The isprime function determines if a number is prime or not by iterating through numbers up to n-1 and checking for divisibility. It initializes a boolean variable result to true and modifies it to false if the number is found to be divisible. The function also prints whether the number is prime or not, which should be avoided. It should only return the boolean result. The test isprime function tests the isprime function with specific numbers and prints the expected result.

Q4b:

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

// Function to factorize a number and return its divisors in a vector
std::vector<int> factorize(int num) {
    std::vector<int> answer; // Vector to store the divisors
    int j;
    for (j = 1; j <= num; j++) {
        if (num % j == 0) // Check if 'j' is a divisor of 'num'
            answer.push_back(j); // Add 'j' to the vector if it's a divisor
    }
    return answer; // Return the vector containing divisors
}

// Function to print the elements of a vector
void print_vector(std::vector<int> v) {
    std::cout << "The factorize vector elements are : " << endl;

    for (int i = 0; i < v.size(); i++)
        std::cout << v.at(i) << ' ' << endl; // Print each element of the vector
}

// Function to test the factorize function with specific numbers
void test_factorize() {
    // Print the factorized vectors for different numbers
    print_vector(factorize(2));
    print_vector(factorize(72));
    print_vector(factorize(196));
}

int main() {
    test_factorize(); // Call the function to test factorization
    return 0;
}
```

As for this code, factorize function computes the divisors of a number and stores them in a vector named answer. print_vector function prints the elements of a given vector. test_factorize function tests the factorize function by factorizing and printing vectors for specific numbers (2, 72, and 196) using print_vector. main function initiates the testing process by calling test_factorize.

This code demonstrates factorization of numbers and prints their divisors using vectors. Each function serves a specific purpose within the code, aiding in testing and displaying the factorization results.

Q4c:

```
#include <iostream>
# include <stdio.h>
# include <math.h>
#include <vector>
using namespace std;

// Function to calculate and return all prime factors of a given number 'n'
std::vector<int> prime_factorize(int n) {
    std::vector<int> answer;

    // Print the number of 2s that divide 'n'
    while (n % 2 == 0) {
        answer.push_back(2); // 2 is a prime factor of 'n'
        n = n / 2; // Divide 'n' by 2 until it's no longer divisible
    }

    // 'n' must be odd at this point, so check for odd factors starting from 3
    for (int i = 3; i <= sqrt(n); i = i + 2) {
        // While 'i' divides 'n', print 'i' as a prime factor and divide 'n'
        while (n % i == 0) {
            answer.push_back(i); // 'i' is a prime factor of 'n'
            n = n / i; // Divide 'n' by 'i' until it's no longer divisible
        }
    }

    // Handling the case when 'n' is a prime number greater than 2
    if (n > 2)
        answer.push_back(n); // Add the remaining 'n' as a prime factor
    return answer; // Return the vector containing prime factors
}


/* Test function for prime factorization */

// Function to print elements of a vector
void print_vector(std::vector<int> v) {
    std::cout << "The prime factorize vector elements are : " << endl;

    for (int i = 0; i < v.size(); i++)
        std::cout << v.at(i) << ' ' << endl; // Print each element of the vector
}

// Function to test the prime_factorize function with specific numbers
void test_prime_factorize() {
    // Print the prime factorized vectors for different numbers
    print_vector(prime_factorize(2));
    print_vector(prime_factorize(72));
    print_vector(prime_factorize(196));
}

int main() {
    test_prime_factorize(); // Call the function to test prime factorization
    return 0;
}
```

Regarding this code, prime_factorize function calculates the prime factors of a number 'n' and stores them in a vector named 'answer'. It first handles factors of 2 separately and then checks odd factors starting from 3 up to the square root of 'n'. The function then returns a vector containing all the prime factors of the given number 'n'. print_vector function prints the elements of a given vector. test_prime_factorize function tests the prime_factorize function with specific numbers and prints the prime factorized vectors. main function initiates the testing process by calling test_prime_factorize.

This code demonstrates the calculation of prime factors for a given number and displays these factors using vectors. Each function serves a specific purpose within the code, aiding in testing and displaying the factorization results.

Q5:

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

// Function to calculate binomial coefficient C(n, k)
int binomialCoeff(int n, int k) {
    int res = 1;
    if (k > n - k)
        k = n - k;

    // Calculate binomial coefficient using nCk formula
    for (int i = 0; i < k; ++i) {
        res *= (n - i);
        res /= (i + 1);
    }

    return res; // Return the calculated binomial coefficient
}

// Function to print the first 'n' lines of Pascal's Triangle
void printPascal(int n) {
    // Iterate through every line and print entries in it
    for (int line = 0; line < n; line++) {
        // Print numbers in each line separated by space
        for (int i = 0; i <= line; i++)
            cout << " " << binomialCoeff(line, i); // Print binomial coefficients
        cout << "\n"; // Move to the next line after printing each line
    }
}

int main() {
    int n;
    cout << "Please Enter a number of rows of Pascal's triangle: " << endl;
    cin >> n; // Input the number of rows for Pascal's Triangle
    printPascal(n); // Call function to print Pascal's Triangle
    return 0;
}
```

As for this code, binomialCoeff function calculates the binomial coefficient C(n, k) using the formula (n!)/(k!(n-k)!). It uses this calculation to fill each element in Pascal's Triangle. printPascal function prints the first 'n' lines of Pascal's Triangle by calling binomialCoeff for each element. In main, the user is prompted to input the number of rows for Pascal's Triangle, and then the printPascal function is called to print the triangle accordingly.

This code generates and prints Pascal's Triangle based on the number of rows input by the user. Each function serves a specific purpose in the process of computing and displaying Pascal's Triangle.