INSTITUTE OF TECHNOLOGY AND MANAGEMENT
SKILLS UNIVERSITY,
KHARGHAR, NAVI MUMBAI

C++ PROGRAMMING LAB

Prepared by:

Name of Student: **Jeevan Naidu**

Roll No: **15**

Batch: 2023-27

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Exp. No	List of Experiment
1	Write a program to find the roots of a quadratic equation.
2	Write a program to calculate the power of a number using a loop.
3	Write a program to check if a given string, is a palindrome.
4	Write a program that simulates a simple ATM machine, allowing users to check their balance, deposit, or withdraw money using a switch statement.
5	Write a program that finds the largest among three numbers using nested if-else statements
6	Write a program that determines the grade of a student based on their marks of 5 subjects using if-else-if ladder.
7	Write a program to find the sum of digits of a number until it becomes a single-digit number.
8	Write a program to print a Pascal's triangle using nested loops.
9	Write a program to calculate the sum of series 1/1! + 2/2! + 3/3! + ... + N/N! using nested loops.
10	Write a program to create an array of strings and display them in alphabetical order.
11	Write a program that checks if an array is sorted in ascending order.
12	Write a program to calculate the sum of elements in each row of a matrix.
13	Write a program to generate all possible permutations of a string.

14	Create a C++ program to print the following pattern: ***** * * * * * * *****
15	Write a C++ program to display the following pattern: 1 232 34543 4567654 34543 232

16	Write a program to creating an inventory management system for a small store. The system should use object-oriented principles in C++. Yourprogram should have the following features:
	• Create a **Product** class that represents a product in the inventory. Each **Product** object should have the following attributes:
	  • Product ID (an integer)
	  • Product Name (a string)
	  • Price (a floating-point number)
	  • Quantity in stock (an integer)
	• Implement a parameterized constructor for the **Product** class to initialize the attributes when a new product is added to the inventory.

17	Write a program to manage student records. Create a class Student with attributes such as name, roll number, and marks. Implement methods for displaying student details, adding new students, and calculating the average marks of all students in the record system.
18	Write a program that implements a basic calculator. Use a class Calculator with methods to perform addition, subtraction, multiplication, and division of two numbers. The program should allow the user to input two numbers and select an operation to perform.

19	Write a program to simulate a simple online shop. Create a class Product with attributes like name, price, and quantity in stock. Implement methods for adding products to the shopping cart, calculating the total cost, and displaying the contents of the cart.
20	Write a program to manage student grades for a classroom. Create a class Student with attributes for student name and an array to store grades. Implement methods for adding grades, calculating the average grade, and displaying the student's name and grades. Use constructors and destructors to initialize and release resources.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 01

Title: Write a program to find the roots of a quadratic equation.

Theory:

This C++ program calculates the roots of a quadratic equation of the form \(ax^2 + bx + c = 0\) using the discriminant \(b^2 - 4ac\). The discriminant determines the nature of the roots.

1. Input:

- The user is prompted to input coefficients \(a\), \(b\), and \(c\) of the quadratic equation.

2. Conditions:

- The program checks if \(a\) is zero. If \(a = 0\), it handles special cases:

- If \(b = 0\) and \(c = 0\), it indicates an identity equation with infinite solutions.

- If \(b = 0\) and \(c \neq 0\), it represents a contradiction with no solution.

- If \(b \neq 0\), it's a linear equation with one root calculated using \(x = -c/b\).

- If \(a \neq 0\), the program proceeds to calculate the discriminant (\(d = b^2 - 4ac\)).

3. Root Calculation:

- If \(b = 0\), it checks for linear equation scenarios.

- If \(c = 0\), it prints \(x = 0\) and another root at \(x = -b/a\).

- If the discriminant (\(d\)) is greater than zero, it calculates and prints distinct real roots.

- If \(d = 0\), it calculates and prints a repeated real root.

- If \(d < 0\), it calculates and prints complex roots.

Code:

```
//Calculate the roots of a quadratic equation using the discriminant.
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    //ax^2+bx+c=0 is the quadratic equation
    //b^2-4ac is discriminant.
```

```
    double d,a,b,c;
    //let d be Discriminant.

    cout<< "Enter the value of 'a': "; //asking user for the input
    cin>>a;                            //initializing it with user input

    cout<< "Enter the value of 'b': "; //asking user for the input
    cin>>b;                            //initializing it with user input

    cout<< "Enter the value of 'c': "; //asking user for the input
    cin>>c;                            //initializing it with user input

    if (a == 0)
    {
        if (b == 0) //if a=0, and also b=0, do...
        {
            if (c == 0) //if all three a = b = c = 0, do...
            {
                cout<< "Infinite solutions (Identity equation)" <<endl; //print
this
            }
            else    // if c!=0 then print the following
            {
                cout<< "No Solution (Contadiction)"<<endl;
            }
        }
        else   //if b!=0 do the following
        {
            double x = -c/b ;  //formula.
            cout << "(Linear Equation) One root at 'x' = "<< x <<endl; //printing
the result
        }
    }
    else //if a != 0 , do the following
    {
        d= (b*b)- 4*a*c ; //formula
        cout<< "Discriminant is: "<< d<< endl; //print the value of discriminant


        if ( b == 0) //when b =0
        {
            if (c == 0) //when b and c both = 0
            {
                cout<<"No solution (Contradiction)"<<endl; //print this.
            }
            else //when c!= 0
            {
                double x = -c/a;   //formula
```

```
                cout<< "(Linear Equation) One root at x = "<<x<<endl; //print
this.
            }
        }
        else if (c == 0) //when c=0
            {
                cout<<"x = 0 "<<endl; //print this and also
                cout << "Another root at x = " << (-b / a) << endl;// print this
            }

        else if (d > 0) //when discriminant is greater than zero then...
        {
            double x1 = (-b + sqrt(d)) / (2 * a); //formula
            double x2 = (-b - sqrt(d)) / (2 * a); //formula
            cout << "Distinct Real Roots: Root 1 = " << x1 << ", Root 2 = " << x2
<< endl; //print the roots of the equation
        }
        else if (d == 0) //when discriminant is = 0 , then...
        {
            double x = -b / (2 * a); //formula
            cout << "Repeated Real Root: x = " << x << endl; //print the root
        }
        else //when dicriminant is lesser than zero ,then...
        {
            double real = -b / (2 * a); //formula
            double imaginary = sqrt(abs(d)) / (2 * a); //formula
            cout << "Complex Roots: Root 1 = " << real << " + " << imaginary <<
"i, Root 2 = " << real << " - " << imaginary << "i" << endl; //print the roots of
the equation
        }
    }

    return 0; //end of this
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                    Code

cd "/Users/jeevan/Documents/the_new_mission/" && g++ 5_quad_roots.cpp -o 5_quad_roots && "/Users/jeevan/Documents/the_new_mission/"5_qua
d_roots
jeevan@Jeevans-MacBook-Air the_new_mission % cd "/Users/jeevan/Documents/the_new_mission/" && g++ 5_quad_roots.cpp -o 5_quad_roots && "/
Users/jeevan/Documents/the_new_mission/"5_quad_roots
Enter the value of 'a': 2
Enter the value of 'b': 3
Enter the value of 'c': 4
Discriminant is: -23
Complex Roots: Root 1 = -0.75 + 1.19896i, Root 2 = -0.75 - 1.19896i
jeevan@Jeevans-MacBook-Air the_new_mission %
```

Output: (screenshot)

Test Case: Any two (screenshot)

Conclusion:

The program provides a comprehensive solution for quadratic equations, handling various scenarios including linear equations, repeated real roots, distinct real roots, and complex roots. It incorporates user input and discriminant-based logic to determine the nature of the roots. The code demonstrates a good understanding of quadratic equations and their solutions.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 02

Title: Write a program to calculate the power of a number using a loop.

Theory:

This C++ program calculates the result of raising a given base to a specified power using a for loop. It follows these steps:

1. Input:

   - The user is prompted to input the base and power of the number.

2. Loop:

   - A for loop is used to iterate from 1 to the specified power.

   - In each iteration, the variable `number` is multiplied by the given `base`.

3. Output:

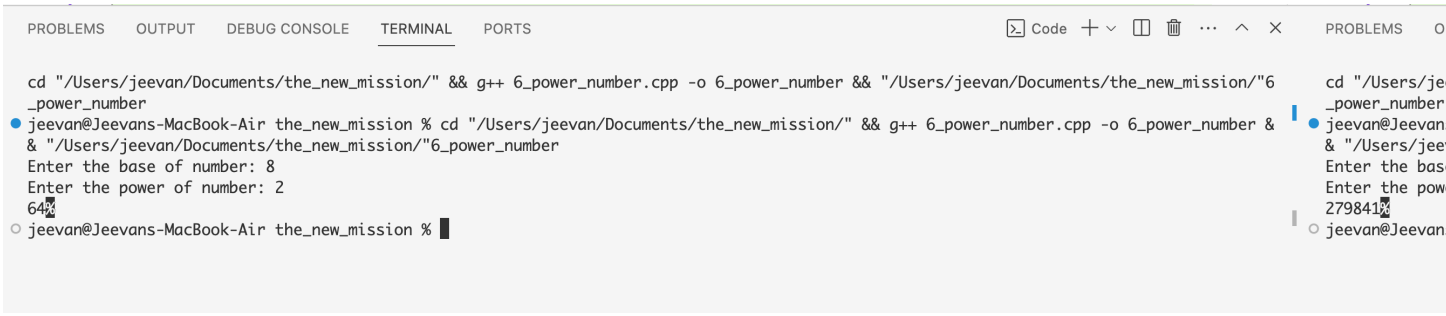
   - The final value of `number` is printed, representing the result of raising the base to the power.

Code:

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

int main()
{
    int base, power, number = 1;

    cout<< "Enter the base of number: ";
    cin>> base;

    cout<< "Enter the power of number: ";
    cin>> power;

    for (int i = 1; i <=power; i++)
    {
        number *= base;
    }
```

```
    cout<< number;
    return 0;
}
```

Output: (screenshot)



Test Case: Any two (screenshot)

Conclusion:

The program efficiently calculates the result of exponentiation using a loop. It demonstrates a basic understanding of iterative processes and is a simple example of how to implement exponentiation in programming. The result is obtained by successively multiplying the base, and the program concludes by printing the final calculated value. This code can be a practical example in a manual to illustrate the concept of exponentiation and the use of loops for iterative calculations.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 03

Title: Write a program to check if a given string, is a palindrome.

Theory:

This C++ program determines whether a given string is a palindrome or not. A palindrome is a word, phrase, number, or other sequence of characters that reads the same forward as backward.

1. Input:

- The user is prompted to input a string.

2. String Length:

- The program calculates the length of the input string using the `size()` function.

3. Palindrome Check:

- A for loop is used to iterate through half of the string.

- In each iteration, it compares the characters from the beginning and end of the string.

- If at any point the characters are not equal, it concludes that the string is not a palindrome and prints the result.

4. Output:

- The program prints whether the given string is a palindrome or not.

Code:

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

int main()
{
    string input;
    cout<<"Enter the String: ";
    cin>> input;

    int length = input.size();

    for (int i = 0; i < length ; i++)
    {
```

```cpp
        if (input[i] == input[length - 1-i])
        {
            cout<< "The given string ( "<< input<< " ) is a Palindrome. ";
            break;
        }
        else
        {
            cout<< "The given string ( "<< input << " ) is not a Palindrome. ";
            break;
        }
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    Code

cd "/Users/jeevan/Documents/the_new_mission/" && g++ 7_string_palindrome.cpp -o 7_string_palindrome && "/Users/jeevan/Documents/the_new_
mission/"7_string_palindrome
jeevan@Jeevans-MacBook-Air the_new_mission % cd "/Users/jeevan/Documents/the_new_mission/" && g++ 7_string_palindrome.cpp -o 7_string_pa
lindrome && "/Users/jeevan/Documents/the_new_mission/"7_string_palindrome
Enter the String: asdf
The given string ( asdf ) is not a Palindrome. %
jeevan@Jeevans-MacBook-Air the_new_mission %
```

Output: (screenshot)

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    Code

cd "/Users/jeevan/Documents/the_new_mission/" && g++ 7_string_palindrome.cpp -o 7_string_palindrome && "/Users/jeevan/Documents/the_new_
mission/"7_string_palindrome
jeevan@Jeevans-MacBook-Air the_new_mission % cd "/Users/jeevan/Documents/the_new_mission/" && g++ 7_string_palindrome.cpp -o 7_string_pa
lindrome && "/Users/jeevan/Documents/the_new_mission/"7_string_palindrome
Enter the String: madam
The given string ( madam ) is a Palindrome. %
jeevan@Jeevans-MacBook-Air the_new_mission %
```

Test Case: Any two (screenshot)

Conclusion:

This program efficiently checks whether a given string is a palindrome by comparing characters from the beginning and end. It correctly uses a loop to iterate through the string and provides a clear output indicating whether the input string is a palindrome or not. This code can serve as an illustrative example in a manual to demonstrate string manipulation and the concept of palindromes in programming.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 04

Title: Write a program that simulates a simple ATM machine, allowing users to check their balance, deposit, or withdraw money using a switch statement.

Theory:

This C++ program simulates a basic interactive ATM system for a hypothetical bank, providing options in multiple languages (English, Hindi, and Marathi). The program structure includes nested switch statements for language selection and subsequent banking operations.

1. Main Menu:

- The program starts with a welcome message and prompts the user to insert their chip card.

- The user is asked to choose a language (1 for English, 2 for Hindi, 3 for Marathi).

2. Language Selection:

- Based on the language choice, the program enters a nested switch statement for the selected language.

- Each language option has its own set of operations: Withdraw Cash, Check Balance, and Deposit.

3. Operations:

- Withdraw Cash:

- The user is prompted to select an account type (Current, Savings, Credit card).

- For each account type, specific withdrawal processes are implemented, including OTP verification and PIN validation.

- Check Balance:

- The user is asked to enter their ATM PIN, and the program displays the account balance if the PIN is correct.

- Deposit:

- The user can deposit a specified amount, and the program assumes a successful deposit.

Code:

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

```
int main() {
    int choice;

    cout << "\n\nWelcome to I bank!\n\n";
    cout << "Please insert your chip card\n\n";

    cout << " (1) Insert your card\n";
    cin >> choice;

    switch (choice) {
        case 1: {
            for (int j = 0; j < 2; j++) {
                cout << "Loading . . .\n\n";
            }

            cout << "We are processing your transaction.\n";
            for (int j = 0; j < 2; j++) {
                cout << "Loading . . .\n\n";
            }

            cout << "Please do not remove your chip card.\nLeave your card inserted during
the entire transaction.\n\n";

            int language;
            cout << "Please choose the language of your choice\n";
            cout << "(1) English\n";
            cout << "(2) हिंदी\n";
            cout << "(3) मराठी\n\n";
            cin >> language;

            switch (language) {
                case 1: {
                    int eng_choice1;
                    cout << "Hello, Customer!\n";
                    cout << "(1) Withdraw Cash\n(2) Check your Balance\n(3) Deposit\n\n";
                    cin >> eng_choice1;

                    switch (eng_choice1) {
                        case 1: {
                            int c1_eng_1;
                            cout << "(1) Current Account\n(2) Savings Account\n(3) Credit
card\n\n";
                            cin >> c1_eng_1;

                            switch (c1_eng_1) {
                                case 1: {
                                    int amount;
                                    int pin, otp = 1234;
                                    int a;

                                    start_withdraw:
                                    cout << "Enter the amount you wish to withdraw: ";
                                    cin >> amount;

                                    cout << "Check OTP on your registered Mobile No.\n";
                                    cout << "If received, press 1 to continue: ";
                                    cin >> a;
```

```
                    if (a != 1) {
                        cout << "Resending OTP...\n";
                        goto start_withdraw;
                    }

                    cout << "Please enter 4-digit OTP received over SMS:
";
                    cin >> otp;

                    if (otp != 1234) {
                        cout << "Invalid OTP. Please try again.\n";
                        goto start_withdraw;
                    }

                    cout << "Please wait while we process your request...
\n";
                    cout << "Please collect your Cash.\n";
                    cout << "Count before leaving.\n";
                    cout << "Thank you.\n";

                    break;
                }

                case 2: {
                    int amount;
                    int pin;

                    start_savings:
                    cout << "Enter the amount you wish to withdraw: ";
                    cin >> amount;

                    if (amount > 5529) {
                        cout << "Insufficient amount.\n";
                        goto start_savings;
                    }

                    here_savings:
                    cout << "Enter your ATM PIN: ";
                    cin >> pin;

                    if (pin != 2023) {
                        cout << "Enter a Valid PIN.\n";
                        goto here_savings;
                    }

                    cout << "Please wait while we process your request...
\n";
                    cout << "Please collect your Cash.\n";
                    cout << "Count before leaving.\n";
                    cout << "Thank you.\n";

                    break;
                }

                default:
                    cout << "Server not found.\n";
                    break;
```

```
                            }
                            break;
                        }

                        case 2: {
                            // Check Balance
                            int pin;

                            cout << "Enter your ATM PIN: ";
                            cin >> pin;

                            if (pin == 2023) // Assuming PIN is 2023 for this example
                            {
                                cout << "Your balance is $1000.00\n"; // Example balance
                            } else {
                                cout << "Invalid PIN. Please try again.\n";
                            }

                            break;
                        }

                        case 3: {
                            // Deposit
                            float depositAmount;
                            cout << "Enter the amount you want to deposit: $";
                            cin >> depositAmount;

                            // Assuming deposit is successful
                            cout << "You have successfully deposited $" << depositAmount
<< ".\n";

                            break;
                        }

                        default:
                            cout << "Server not found.\n";
                            break;
                    }
                    break;
                }

                case 2: {
                    int hind_choice1;
                    cout << "Hello, Customer!\n";
                    cout << "(1) नकदी निकासी\n(2) Check your Balance\n(3) Deposit\n\n";
                    cin >> hind_choice1;

                    switch (hind_choice1) {
                        case 1: {
                            int c1_hind_1;
                            cout << "(1) चालू\n(2) बचत\n(3) ऋण\n\n";
                            cin >> c1_hind_1;

                            switch (c1_hind_1) {
                                case 1: {
                                    int amount;
                                    int pin, otp = 1234;
                                    int a;
```

```
                start_withdraw_hindi:
                cout << "कृपया राशि दर्ज करें: ";
                cin >> amount;

                cout << "अपने पंजीकृत मोबाइल नंबर पर OTP देखें।\n";
                cout << "यदि प्राप्त हुआ है, तो आगे बढ़ने के लिए 1 दबाएं: ";
                cin >> a;

                if (a != 1) {
                    cout << "OTP पुनः भेज रहे हैं...\n";
                    goto start_withdraw_hindi;
                }

                cout << "कृपया SMS पर प्राप्त 4-अंकीय OTP दर्ज करें: ";
                cin >> otp;

                if (otp != 1234) {
                    cout << "अमान्य OTP। कृपया पुनः प्रयास करें।\n";
                    goto start_withdraw_hindi;
                }

                cout << "कृपया प्रतीक्षा करें, हम आपके अनुरोध का प्रसंस्करण करत
आहोत...\n";
                cout << "कृपया अपने नकदी ले लें \n";
                cout << "जाने से पहले गिन लें।\n";
                cout << "धन्यवाद ! \n";

                break;
            }

            case 2: {
                int amount;
                int pin;

                start_savings_hindi:
                cout << "कृपया निकासी की राशि दर्ज करें: ";
                cin >> amount;

                if (amount > 5529) {
                    cout << "राशि अपर्याप्त है।\n";
                    goto start_savings_hindi;
                }

                here_savings_hindi:
                cout << "अपना एटीएम पिन दर्ज करें: ";
                cin >> pin;

                if (pin != 2023) {
                    cout << "कृपया वैध पिन दर्ज करें।\n";
                    goto here_savings_hindi;
                }

                cout << "कृपया प्रतीक्षा करें, हम आपके अनुरोध का प्रसंस्करण करत
आहोत...\n";
                cout << "कृपया अपने नकदी ले लें \n";
```

```
                                cout << "जाने से पहले गिन लें।\n";
                                cout << "धन्यवाद ! \n";

                                break;
                            }

                            default:
                                cout << "सर्वर नहीं मिला।\n";
                                break;
                        }
                        break;
                    }

                    case 2: {
                        // Check Balance in Hindi
                        int pin;

                        cout << "आपका एटीएम पिन दर्ज करें: ";
                        cin >> pin;

                        if (pin == 2023) // Assuming PIN is 2023 for this example
                        {
                            cout << "आपका शेष $1000.00 है।\n"; // Example balance
                        } else {
                            cout << "अमान्य पिन। कृपया पुनः प्रयास करें।\n";
                        }

                        break;
                    }

                    case 3: {
                        // Deposit in Hindi
                        float depositAmount;
                        cout << "आपकी जमा करने की राशि दर्ज करें: $";
                        cin >> depositAmount;

                        // Assuming deposit is successful
                        cout << "आपने सफलतापूर्वक $" << depositAmount << " जमा किया है।\n";

                        break;
                    }

                    default:
                        cout << "अमान्य विकल्प।\n";
                        break;
                }
                break;
            }

            case 3: {
                int mar_choice1;
                cout << "Hello, Customer!\n";
                cout << "(1) कॅश व्यत्यास\n(2) Check your Balance\n(3) Deposit\n\n";
                cin >> mar_choice1;

                switch (mar_choice1) {
                    case 1: {
```

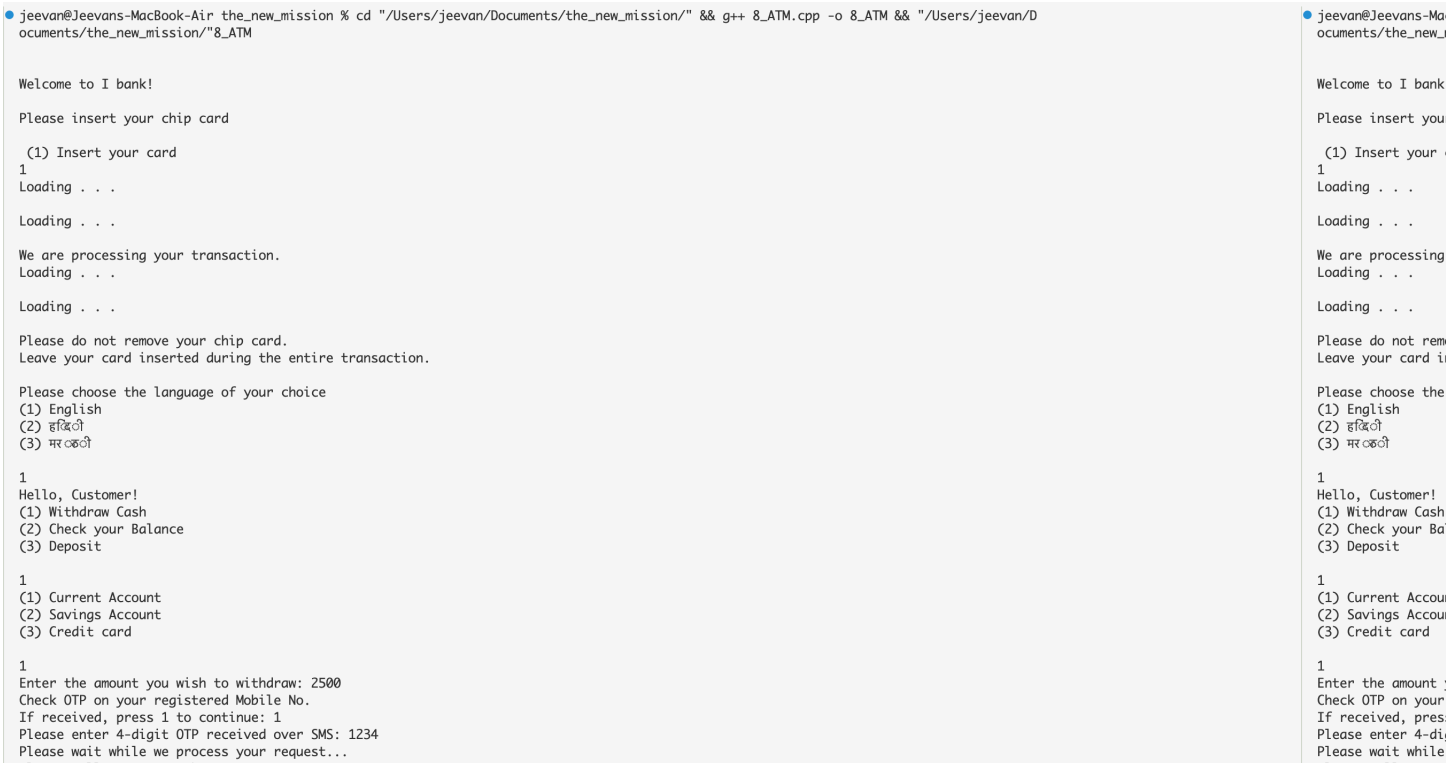
```
                        int c1_mar_1;
                        cout << "(1) चालू\n(2) बचत\n(3) कर्ज\n\n";
                        cin >> c1_mar_1;

                        switch (c1_mar_1) {
                            case 1: {
                                int amount;
                                int pin, otp = 1234;
                                int a;

                                start_withdraw_marathi:
                                cout << "कृपया रक्कम नोंदवा:  ";
                                cin >> amount;

                                cout << "आपल्या नोंदणीकृत मोबाइल नंबरवरून OTP पहा.\n";
                                cout << "जर प्राप्त झालं असलं तर आगल्या कदानासाठी १ दाबा: ";
                                cin >> a;

                                if (a != 1) {
                                    cout << "OTP पुनः पाठवत आहोत...\n";
                                    goto start_withdraw_marathi;
                                }

                                cout << "कृपया SMS द्वारे प्राप्त केलेला ४-अंकीय OTP प्रविष्ट करा: ";
                                cin >> otp;

                                if (otp != 1234) {
                                    cout << "अवैध OTP। कृपया पुनः प्रयास करा.\n";
                                    goto start_withdraw_marathi;
                                }

                                cout << "कृपया थांबा, आम्ही तुमचा विनंती प्रसंस्करण करीत आहोत...
\n";
                                cout << "कृपया आपले कॅश घ्या.\n";
                                cout << "जाण्यापूर्वी गिणा.\n";
                                cout << "धन्यवाद ! \n";

                                break;
                            }

                            case 2: {
                                int amount;
                                int pin;

                                start_savings_marathi:
                                cout << "कृपया रक्कम नोंदवा: ";
                                cin >> amount;

                                if (amount > 5529) {
                                    cout << "रक्कम अपर्याप्त आहे.\n";
                                    goto start_savings_marathi;
                                }

                                here_savings_marathi:
                                cout << "आपला एटीएम पिन प्रविष्ट करा: ";
                                cin >> pin;
```

```
                                    if (pin != 2023) {
                                        cout << "कृपया वैध पिन प्रविष्ट करा.\n";
                                        goto here_savings_marathi;
                                    }

                                    cout << "कृपया थांबा, आम्ही तुमचा विनंती प्रसंस्करण करीत आहोत...
\n";
                                    cout << "कृपया आपले कॅश घ्या.\n";
                                    cout << "जाण्यापूर्वी गिणा.\n";
                                    cout << "धन्यवाद ! \n";

                                    break;
                                }

                                default:
                                    cout << "सर्वर सापडलं नाही.\n";
                                    break;
                            }
                            break;
                        }

                        case 2: {
                            // Check Balance in Marathi
                            int pin;

                            cout << "आपला एटीएम पिन प्रविष्ट करा: ";
                            cin >> pin;

                            if (pin == 2023) // Assuming PIN is 2023 for this example
                            {
                                cout << "आपल्याच शिल्लक आहे $1000.00.\n"; // Example balance
                            } else {
                                cout << "अवैध पिन. कृपया पुनः प्रयास करा.\n";
                            }

                            break;
                        }

                        case 3: {
                            // Deposit in Marathi
                            float depositAmount;
                            cout << "जमा करण्याची रक्कम प्रविष्ट करा: $";
                            cin >> depositAmount;

                            // Assuming deposit is successful
                            cout << "आपल्याना सफलतेचा विश्वास आहे $" << depositAmount << ".\n";

                            break;
                        }

                        default:
                            cout << "सर्वर सापडलं नाही.\n";
                            break;
                    }
                    break;
                }
```

```
                default:
                    cout << "Server not found.\n";
                    break;
            }

            return 0;
        }

        default:
            cout << "Invalid Choice\n";
            break;
    }

    return 0;
}
```

```
cd "/Users/jeevan/Documents/the_new_mission/" && g++ 8_ATM.cpp -o 8_ATM && "/Users/jeevan/Documents/the_new_mission/"8_ATM
jeevan@Jeevans-MacBook-Air the_new_mission % cd "/Users/jeevan/Documents/the_new_mission/" && g++ 8_ATM.cpp -o 8_ATM && "/Users/jeevan/D
ocuments/the_new_mission/"8_ATM


Welcome to I bank!

Please insert your chip card

 (1) Insert your card
1
Loading . . .

Loading . . .

We are processing your transaction.
Loading . . .

Loading . . .

Please do not remove your chip card.
Leave your card inserted during the entire transaction.

Please choose the language of your choice
(1) English
(2) हिंदी
(3) मराठी

1
Hello, Customer!
(1) Withdraw Cash
(2) Check your Balance
(3) Deposit

2
Enter your ATM PIN: 1234
Invalid PIN. Please try again.
jeevan@Jeevans-MacBook-Air the_new_mission %
```

Output: (screenshot)



Test Case: Any two (screenshot)

Conclusion:

- The program provides a simulated experience of an ATM system with language selection and basic banking operations.

- It handles various scenarios such as withdrawal with OTP, withdrawal with PIN validation, balance check, and deposit.

- The use of nested switch statements allows for language-specific and operation-specific handling.

- This code can be part of a practical manual to demonstrate the implementation of a simple ATM system in C++ with user interactions, input validation, and language customization.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 05

Title: Write a program that finds the largest among three numbers using nested if-else statements

Theory:

1. Header Files:

- `#include <iostream>`: This line includes the input-output stream header file, allowing the use of functions like `cout` and `cin` for input and output operations.

2. Namespace:

- `using namespace std;`: This line avoids the need to use the `std::` prefix before standard C++ library functions.

3. Main Function:

- `int main()`: The starting point of execution for the program. It returns an integer value to the operating system.

4. Variable Declaration:

- `int a, b, c;`: Three integer variables (`a`, `b`, and `c`) are declared to store user input.

5. User Input:

- `cout << "Enter the First number: ";`: Displays a prompt for the user to enter the first number.
- `cin >> a;`: Takes user input and stores it in variable `a`. Similar steps are repeated for `b` and `c`.

6. Conditional Statements:

- The program uses `if-else if-else` statements to compare the values of `a`, `b`, and `c` to determine the greatest number.
- If `a` is greater than both `b` and `c`, it prints that `a` is the greatest.
- If `b` is greater than both `a` and `c`, it prints that `b` is the greatest.
- Otherwise, it concludes that `c` is the greatest.

7. Output:

- The program prints the result using `cout` statements.

8. Return Statement:

- `return 0;`: Indicates the successful execution of the program to the operating system.

Code:

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

int main()
{
    int a,b,c;

    cout<<"Enter the First number: ";
    cin>> a;

    cout<<"Enter the Second number: ";
    cin>> b;

    cout<< "Enter the Third number: ";
    cin>> c;

    if (a>b && a>c)
    {
        cout<< a << " is greatest among all i.e.(" << b << " and " << c << "). ";
    }
    else if (b> a && b>c)
    {
        cout<< b << " is greatest among all i.e.(" << a << " and " << c << "). ";
    }
    else
    {
        cout<< c << " is greatest among all i.e.(" << a << " and " << b << "). ";
    }
    return 0;
}
```

```
cd "/Users/jeevan/Documents/the_new_mission/" && g++ 10_largest_of_three.cpp -o 10_largest_of_three && "/Users/jeevan/Documents/the_new_mission/"10_largest_of_three
jeevan@Jeevans-MacBook-Air the_new_mission % cd "/Users/jeevan/Documents/the_new_mission/" && g++ 10_largest_of_three.cpp -o 10_largest_of_three && "/Users/jeevan/Documents/
the_new_mission/"10_largest_of_three
Enter the First number: 2
Enter the Second number: 3
Enter the Third number: 1
3 is greatest among all i.e.(2 and 1). %
jeevan@Jeevans-MacBook-Air the_new_mission %
```

Output: (screenshot)

```
cd "/Users/jeevan/Documents/the_new_mission/" && g++ 10_largest_of_three.cpp -o 10_largest_of_three && "/Users/jeevan/Documents/the_new_mission/"10_largest_of_three
jeevan@Jeevans-MacBook-Air the_new_mission % cd "/Users/jeevan/Documents/the_new_mission/" && g++ 10_largest_of_three.cpp -o 10_largest_of_three && "/Users/jeevan/Documents/
the_new_mission/"10_largest_of_three
Enter the First number: 7
Enter the Second number: 2
Enter the Third number: 13
13 is greatest among all i.e.(7 and 2). %
jeevan@Jeevans-MacBook-Air the_new_mission %
```

Test Case: Any two (screenshot)

```
cd "/Users/jeevan/Documents/the_new_mission/" && g++ 10_largest_of_three.cpp -o 10_largest_of_three && "/Users/jeevan/Documents/the_new_mission/"10_largest_of_three
jeevan@Jeevans-MacBook-Air the_new_mission % cd "/Users/jeevan/Documents/the_new_mission/" && g++ 10_largest_of_three.cpp -o 10_largest_of_three && "/Users/jeevan/Documents/
the_new_mission/"10_largest_of_three
Enter the First number: 9
Enter the Second number: 81
Enter the Third number: 77
81 is greatest among all i.e.(9 and 77). %
jeevan@Jeevans-MacBook-Air the_new_mission %
```

Conclusion:

This program takes three integer inputs from the user and determines the greatest among them using conditional statements. It demonstrates the use of basic input/output operations and conditional branching in C++. The conclusion is printed based on the comparison of the three numbers. The user is informed about which number is the greatest among the entered values.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 06

Title: Write a program that determines the grade of a student based on their marks of 5 subjects using if-else-if ladder.

Theory:

1. Variable Declaration:

- `double sum = 0.0;`: Initializes the variable `sum` to store the total marks.
- `double n;`: Represents the variable `n` to temporarily store the marks of each subject.
- `double marks[5] = {0};`: Initializes an array `marks` to store the marks of five subjects.

2. User Input Loop:

- A `for` loop iterates five times, prompting the user to input marks for each subject.
- The use of `goto` is seen to handle the case where the user enters marks greater than 100, prompting for a valid score.

3. Total and Average Calculation:

- The program calculates the total marks (`sum`) and the average marks (`average`) based on the input.

4. Grade Assignment:

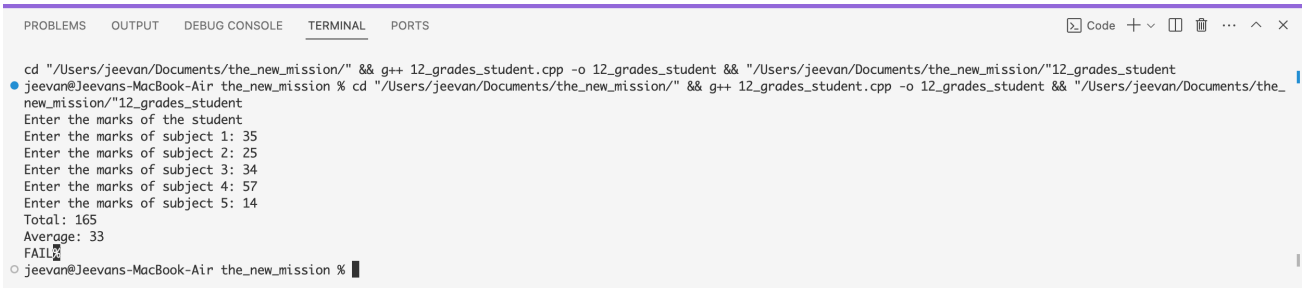
- The program uses `if-else` statements to assign a grade based on the average marks.
- If the average is greater than or equal to 70, it assigns the grade 'A'.
- If the average is between 60 and 69, it assigns the grade 'B'.
- If the average is between 45 and 59, it assigns the grade 'C'.
- Otherwise, it indicates a failing grade.

Code:

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

int main()
{
    double sum=0.0;
```

```
    double n;
    double marks[5]={0};
    cout<< "Enter the marks of the student"<<endl;

    for (int i = 0; i < 5; i++)
    {
       here:
       cout<< "Enter the marks of subject " << i+1 <<": ";
       cin >> n;

       if (n > 100)
       {
        cout << "Enter the valid score of the student"<<endl;
        goto here;
       }

       marks[i] = n;
       sum += marks [i];
    }

    cout<< "Total: " << sum<<endl;

    double average = (sum/5) ;
    cout<<"Average: " <<average <<endl;

    if (average>= 70)
    {
        cout<< "GRADE: A";
    }
    else if (average>=60)
    {
        cout<< "GRADE: B";
    }
    else if (average>=45)
    {
        cout<< "GRADE: C";
    }
    else
    {
        cout<< "FAIL";
    }
   return 0;
}
```

Output: (screenshot)



Test Case: Any two (screenshot)

Conclusion:

This C++ program is designed to collect marks for five subjects from a student, calculate the total and average marks, and assign a grade based on the average. The program includes a loop to ensure that valid marks (0 to 100) are entered for each subject. If an invalid mark is entered (greater than 100), the program prompts the user to enter a valid score using the `goto` statement.

After gathering all the input, the program calculates the total and average marks. Finally, based on the average, it determines the student's grade and prints the result. The use of conditional statements (`if-else`) makes the grade assignment dynamic, depending on the calculated average.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 07

Title: Write a program to find the sum of digits of a number until it becomes a single-digit number.

Theory:

1. Variable Declaration:

- `int sum = 0;`: Initializes the variable `sum` to store the sum of individual digits.

- `string num;`: Represents a string variable `num` to store the user input.

2. User Input:

- `cout<< "Enter the number: ";`: Prompts the user to enter a number.

- `cin>> num;`: Takes the user input and stores it in the string variable `num`.

3. String Length Calculation:

- `int length = num.length();`: Determines the length of the input string using the `length()` function.

4. Digit Sum Calculation:

- A `for` loop iterates through each character in the string, converting it to an integer and adding it to the `sum`.

5. Output:

- Displays the length of the string and the sum of its digits.

6. Conversion to String:

- `string new_num = to_string(sum);`: Converts the integer `sum` to a string, storing it in `new_num`.

7. Length of New String:

- `int len = log10(sum) + 1;`: Calculates the length of the new string using the logarithmic function.

8. Conditional Statement:

- `if (len > 1)`: Checks if the length of the new string is greater than 1.

9. Nested Loop for Second Sum:

- If the length is greater than 1, a new `for` loop iterates through each character in the new string, converting and adding its digits to a new sum (`new_sum`).

Code:

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

int main()
{
    int sum=0;
    string num;

    cout<< "Enter the number: ";
    cin>> num;

    int length = num.length();
    cout<<"Length of the string: "<< length<<endl<<endl;

    for (int i = 0; i <length; i++)
    {
        sum += (num[i] - '0');
    }
    cout<<"sum of the number: "<< sum<< endl<<endl;


    string new_num= to_string(sum);
    cout<< new_num<<endl;

    int len = log10(sum) + 1;
    cout<< len<< endl;

    if (len > 1)
    {
        int new_sum=0;

        for (int i = 0; i < len; i++)
        {
            new_sum += (new_num[i] - '0');
        }
        cout<<new_sum<< endl<<endl;
    }

    return 0;
}
```

Output: (screenshot)

Test Case: Any two (screenshot)

Conclusion:

This C++ program takes a number as a string input, calculates the sum of its individual digits, and performs additional operations based on the length of the resulting sum. It demonstrates several concepts:

- String Manipulation: The program uses the `length()` function to find the length of the input string and `to_string()` to convert an integer to a string.

- Looping: It utilizes `for` loops to iterate through characters in the input string and the newly created string.

- Type Conversion: The program converts characters to integers (`num[i] - '0'`) and converts an integer to a string (`to_string(sum)`).

- Mathematical Operation: It uses the `log10()` function to calculate the length of an integer.

- Conditional Statements: The program employs an `if` statement to conditionally execute the second loop based on the length of the new string.

Overall, the program illustrates string manipulation, mathematical operations, and control flow in C++. The use of loops and conditional statements enhances the program's flexibility in handling different scenarios based on user input.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 08

Title: Write a program to print a Pascal's triangle using nested loops.

Theory:

1. Function Definition (Recursion):

- `int fact(int n)`: Defines a recursive function `fact` to calculate the factorial of a given number `n`. The base case checks if `n` is 0 or 1, returning 1 in those cases.

2. Nested Loops:

- Two nested `for` loops are used to iterate over the rows and columns of Pascal's Triangle.

3. Space Printing:

- The first nested loop (`for (int j = 1; j <= 5 - i; j++)`) prints spaces to format the output in a triangular pattern.

4. Coefficient Calculation:

- The second nested loop (`for (int k = 0; k <= i; k++)`) calculates and prints the binomial coefficient for each position in the triangle using the formula `C(n, k) = n! / (k! * (n - k)!)`.

5. Factorial Function Usage:

- The `fact` function is used to calculate factorials, aiding in the computation of binomial coefficients.

Code:

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

int fact(int n) {
    if (n == 0 || n == 1) {
        return 1;
    }
    return n * fact(n - 1);
}


int main()
{
    for (int i = 0; i <5; i++)
    {
        for (int j = 1; j <= 5-i; j++)
        {
            cout<< " ";
        }
        for (int k = 0; k <=i; k++)
        {
            cout<< fact(i)/(fact(k)*fact(i-k)) << " ";
        }
        cout<<endl;
    }
    return 0;
}
```

Output: (screenshot)

Test Case: Any two (screenshot)

Conclusion:

This C++ program utilizes the concept of Pascal's Triangle to generate and display the first five rows. It demonstrates the use of functions (including recursion) to calculate factorials and the application of nested loops for pattern printing. The program showcases a mathematical concept (binomial coefficients) and effectively translates it into a visual.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 09

Title: Write a program to calculate the sum of series 1/1! + 2/2! + 3/3! + ... + N/N! using nested loops.

Theory:

1. Function Definition (Recursion):

- `double fact(int n)`: Defines a recursive function `fact` to calculate the factorial of a given number `n`. The base case checks if `n` is 0 or 1, returning 1 in those cases.

2. User Input:

- `double n;`: Declares a variable to store the user input for the upper limit of the series.
- `cout << "Enter till where the series is intended: ";`: Prompts the user to enter the limit of the series.
- `cin >> n;`: Takes user input and stores it in the variable `n`.

3. Series Summation:

- The program uses a `for` loop to iterate from 1 to the user-specified limit (`n`).
- Inside the loop, it calculates each term of the series (`i / fact(i)`) and adds it to the running sum.

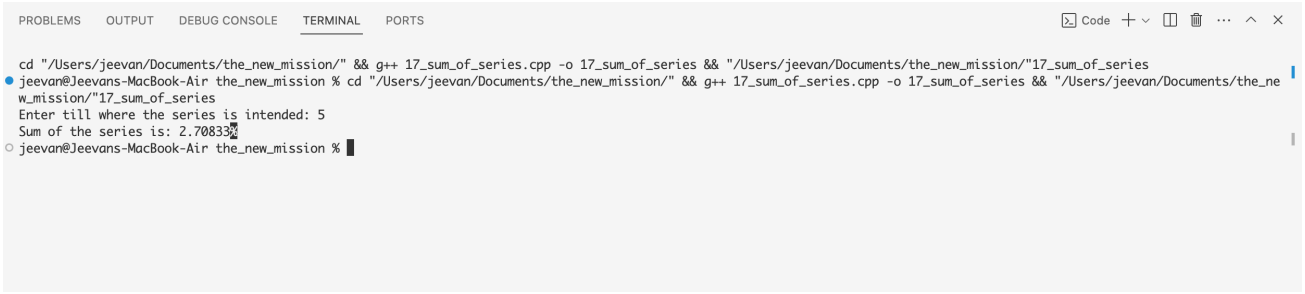
4. Output:

- `cout << "Sum of the series is: " << sum;`: Displays the final sum of the series.

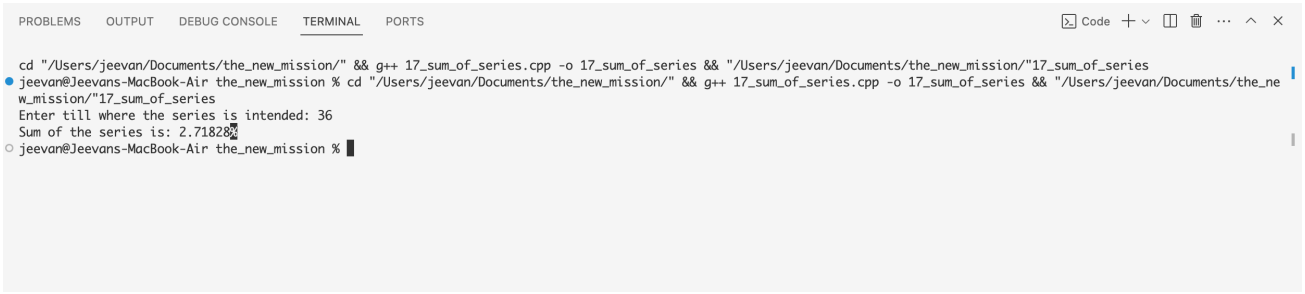
Code:

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

double fact(int n) {
    if (n == 0 || n == 1) {
        return 1;
    }
    return n * fact(n - 1);
```

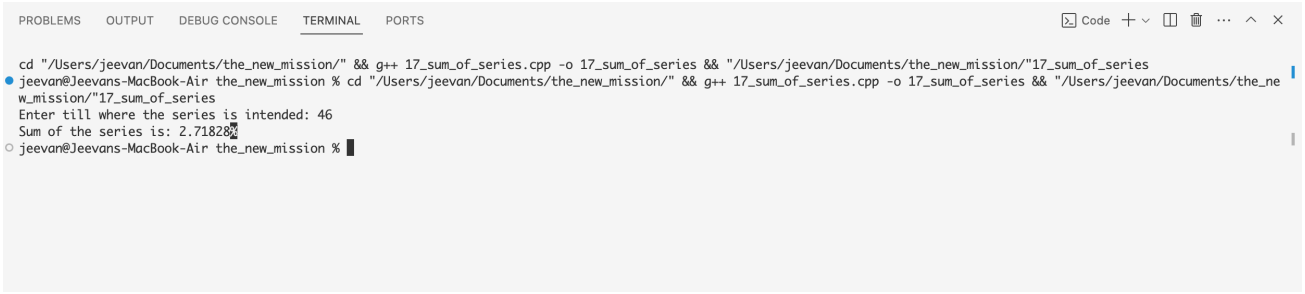
```
}

int main() {
    double n;
    cout << "Enter till where the series is intended: ";
    cin >> n;

    double sum = 0.0;
//show

    for (int i = 1; i <= n; i++) {
        sum += i / fact(i);
    }

    cout << "Sum of the series is: " << sum;

    return 0;
}
```



Output: (screenshot)



Test Case: Any two (screenshot)



Conclusion:

This C++ program implements a mathematical series calculation, demonstrating the use of functions (including recursion) to compute factorials and a loop to sum the series. The series involves terms of the form `i / i!`, where `i` ranges from 1 to the user-specified limit.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 10

Title: Write a program to create an array of strings and display them in alphabetical order.

Theory:

1. User Input:

   - `int n;`: Declares a variable to store the number of strings.

   - `cin >> n;`: Takes user input for the number of strings.

2. String Array:

   - `string a[n];`: Declares an array to store the input strings.

3. Input Loop:

   - A `for` loop is used to input strings based on the user-specified count.

4. Bubble Sort for Strings:

   - Nested `for` loops implement the Bubble Sort algorithm to sort the strings in alphabetical order.

5. String Comparison and Swapping:

   - Strings are compared using `if (a[j] > a[j + 1])` and swapped if necessary to achieve ascending order.

6. Display Original and Sorted Strings:

   - Two separate `for` loops are used to display the original and sorted strings.

Code:

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

int main()
```

```cpp
{
    int n;
    cout << "Enter the no. of strings you want: ";
    cin >> n;

    string a[n];

    for (int i = 0; i < n; i++)
    {
        cout << "input for " << i + 1 << " :" << endl;
        cin >> a[i];
    }

    cout << endl;

    for (int i = 0; i < n; i++)
    {
        cout << a[i] << " ";
    }
    cout << endl;


    for (int i = 0; i < n-1; i++)
    {
        for (int j = 0; j < n-i-1; j++)
        {
            if (a[j] > a[j+1])
            {
                string store = a[j];
                a[j] = a[j+1];
                a[j+1] = store;
            }
        }
    }
    cout<<endl;
    cout << "Strings in alphabetical order:" << endl<<endl;;
    for (int i = 0; i < n; i++)
    {
        cout << a[i] << " ";
    }

    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    Code

cd "/Users/jeevan/Documents/the_new_mission/" && g++ 18_alphabetical_order.cpp -o 18_alphabetical_order && "/Users/jeevan/Documents/the_new_mission/"18_alphabetical_order
jeevan@Jeevans-MacBook-Air the_new_mission % cd "/Users/jeevan/Documents/the_new_mission/" && g++ 18_alphabetical_order.cpp -o 18_alphabetical_order && "/Users/jeevan/Documents/the_new_mission/"18_alphabetical_order
Enter the no. of strings you want: 2
input for 1 :
hello
input for 2 :
btech

hello btech

Strings in alphabetical order:

btech hello %
jeevan@Jeevans-MacBook-Air the_new_mission %
```

Output: (screenshot)

Test Case: Any two (screenshot)

Conclusion:

This C++ program demonstrates the input, sorting, and display of strings in alphabetical order. It uses an array to store the input strings and applies the Bubble Sort algorithm for sorting. The user is prompted to enter the number of strings, and then each string is input one by one.

After displaying the original order of strings, the program sorts them using the Bubble Sort algorithm, and the sorted strings are displayed.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 11

Title: Write a program that checks if an array is sorted in ascending order.

Theory:

1. Dynamic Array Creation:

   - `int size;`: Declares a variable to store the size of the array.

   - `cin >> size;`: Takes user input for the size of the array.

   - `int arr[size];`: Creates an array with a size based on user input.

2. Input Loop:

   - A `for` loop is used to input integers into the array based on the user-specified size.

3. Bubble Sort Algorithm:

   - Nested `for` loops implement the Bubble Sort algorithm to sort the array in ascending order.

   - The algorithm compares adjacent elements and swaps them if they are in the wrong order.

4. Temporary Variable for Swapping:

   - A temporary variable (`int temp`) is used for swapping elements during the Bubble Sort.

5. Displaying Sorted Array:

   - Another `for` loop is used to display the sorted array after the sorting process.

Code:

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

int main()
{
    int size;
    cout << "Enter the size of the array: ";
    cin >> size;
```

```
    int arr[size];

    cout << "Enter " << size << " integers: ";
    for (int i = 0; i < size; i++)
    {
        cin >> arr[i];
    }


    for (int i = 0; i < size-1; i++)
    {
        for (int j = 0; j < size-i-1; j++)
        {
            if (arr[j] > arr[j+1])
            {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }

    cout << "Sorted array in ascending order: ";
    for (int i = 0; i < size; i++)
    {
        cout << arr[i] << " ";
    }

    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    Code

cd "/Users/jeevan/Documents/the_new_mission/" && g++ 20_ascending_order.cpp -o 20_ascending_order && "/Users/jeevan/Documents/the_new_mission/"20_ascending_order
jeevan@Jeevans-MacBook-Air the_new_mission % cd "/Users/jeevan/Documents/the_new_mission/" && g++ 20_ascending_order.cpp -o 20_ascending_order && "/Users/jeevan/Documents/th
e_new_mission/"20_ascending_order
Enter the size of the array: 3
Enter 3 integers: 54
23
12
Sorted array in ascending order: 12 23 54
jeevan@Jeevans-MacBook-Air the_new_mission %
```

Output: (screenshot)

Test Case: Any two (screenshot)

Conclusion:

This C++ program demonstrates the use of dynamic array creation, user input, and the Bubble Sort algorithm for sorting integers in ascending order. The user is prompted to enter the size of the array and then input the integers. The Bubble Sort algorithm is applied to arrange the elements in ascending order.

The program then displays the sorted array. This example showcases fundamental concepts in C++, including dynamic array creation, loops, user input, and a basic sorting algorithm.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 12

Title: Write a program to calculate the sum of elements in each row of a matrix.

Theory:

1. Constant Variables:

- `const int ROWS = 3;` and `const int COLS = 3;`: Declares constant variables for the number of rows and columns in the matrix.

2. 2D Array Initialization:

- `int matrix[ROWS][COLS]`: Declares and initializes a 3x3 matrix.

3. Nested Loops for Matrix Traversal:

- Two nested `for` loops are used to iterate through each row and column of the matrix.

4. Row Sum Calculation:

- `int row_sum = 0;`: Initializes a variable to store the sum of the current row.
- `row_sum += matrix[i][j];`: Adds each element of the current row to the row_sum.

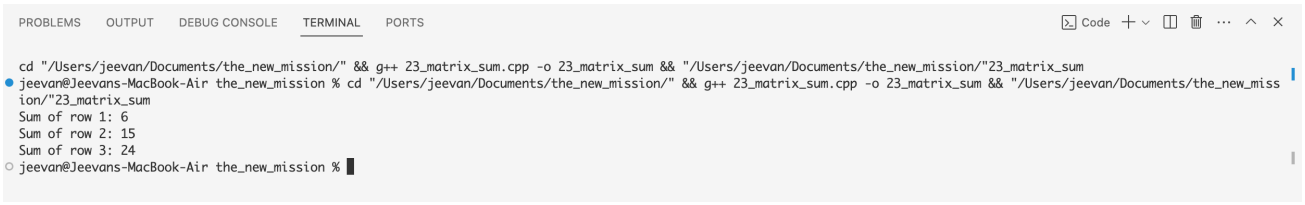
5. Displaying Row Sums:

- `cout << "Sum of row " << i + 1 << ": " << row_sum << endl;`: Displays the sum of each row.

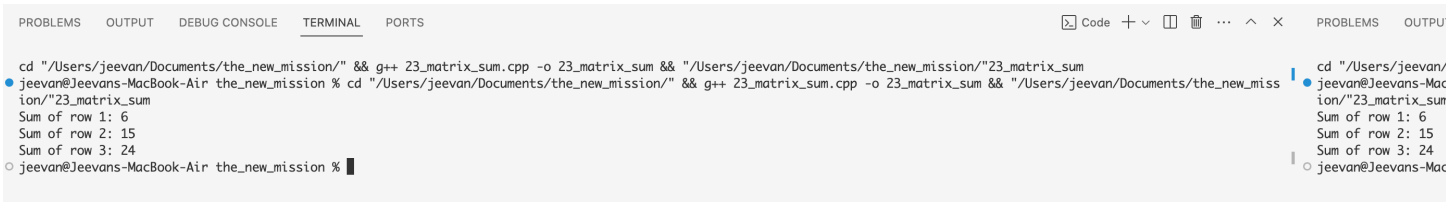
Code:

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

const int ROWS = 3;
const int COLS = 3;

int main() {
    int matrix[ROWS][COLS] = {{1, 2, 3},
                              {4, 5, 6},
                              {7, 8, 9}};
```

```cpp
    for (int i = 0; i < ROWS; i++) {
        int row_sum = 0;
        for (int j = 0; j < COLS; j++) {
            row_sum += matrix[i][j];
        }
        cout << "Sum of row " << i + 1 << ": " << row_sum << endl;
    }

    return 0;
}
```



Output: (screenshot)



Test Case: Any two (screenshot)

Conclusion:

This C++ program demonstrates the calculation of row sums for a 3x3 matrix. It uses nested loops to traverse the matrix, calculates the sum of each row, and displays the results. The use of constant variables for the number of rows and columns enhances code readability and maintainability.

This example showcases fundamental concepts in C++ programming, including array manipulation, nested loops, and basic arithmetic operations. It provides a simple illustration of matrix operations and serves as an introductory example for handling 2D arrays in C++.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 13

Title: Write a program to generate all possible permutations of a string.

Theory:

1. Factorial Function:

- `int fact(int n)`: Defines a recursive function to calculate the factorial of a given number.

2. Recursive Permutation Generation:

- `void generatePermutations(string str, int l, int r, int& count)`: Recursive function to generate all permutations of a string.

3. Swap Function:

- `swap(str[l], str[i]);`: Swaps characters in the string during permutation generation.

4. User Input:

- `string input;`: Declares a variable to store the user-entered string.

5. Displaying Information:

- Displays the length of the input string and the total number of permutations.
- `cout << count << ": " << str << endl;`: Displays each permutation along with a count.

Code:

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

int fact(int n) {
    if (n == 0 || n == 1) {
        return 1;
    }
    return n * fact(n - 1);
}

void generatePermutations(string str, int l, int r, int& count)
{
```

```
    if (l == r)
    {
        cout << count << ": " << str << endl;
        count++;
    }
    else
    {
        for (int i = l; i <= r; i++)
        {
            swap(str[l], str[i]);
            generatePermutations(str, l + 1, r, count);
            swap(str[l], str[i]);
        }
    }
}
int main()
{
    string input;
    cout<< "Enter the string: ";
    cin>> input;

    int length= input.length();
    cout<<endl<<"Length of the string: "<< length<< endl;

    cout<<endl<< "Possible permutations in the string: "<< fact(length)
<< endl;

    //int a=1;

    int count = 1;

    cout <<endl<< "Permutations:" << endl;
    generatePermutations(input, 0, input.length() - 1, count);

    return 0;


}
```

Output: (screenshot)

Test Case: Any two (screenshot)

Conclusion:

This C++ program efficiently generates and displays all permutations of a given string using a recursive algorithm. It showcases the use of recursive functions, string manipulation, and mathematical concepts such as factorial calculation. The program allows users to input a string and then prints information about the string, including its length and the total number of permutations. Each permutation is displayed along with a count.

This example provides insight into recursive algorithms and their application in solving combinatorial problems. It serves as an educational tool for understanding permutation generation and demonstrates how C++ can be used to implement such algorithms.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 14

Title: Create a C++ program to print the following pattern:

```
*****
* *
* *
* *
*****
```

Theory:

1. Nested Loops:

   - Two nested `for` loops are used to control the rows and columns of the pattern.

2. Conditional Statements:

   - `if (i < 5)`: Checks if the current row is not the last row to print spaces accordingly.
   - `if (i == j || i == 5)`: Prints '@' symbols in a diagonal pattern and also on the last row.

3. Pattern Generation:

   - The program generates a pattern using '@' symbols, creating a diagonal line and a staircase-like structures

Code:

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

int main()
{
    // for (int l = 0; l <= 3; l++)
    // {
    //     cout<< " $ ";
    // }
    for (int k = 1; k <= 5; k++)
    {
        cout<< " @ ";
    }
    cout<< endl;
    for (int i = 1; i <= 5; i++)
    {
        if (i < 5)
        {
```

```
            cout<< " @          ";
        }

        for (int j = 1; j <= i; j++)
        {
            if (i==j || i == 5)
            {
                cout<< " @ ";
            }
        }
        cout<< endl;
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    Code

cd "/Users/jeevan/Documents/the_new_mission/" && g++ 25_pattern.cpp -o 25_pattern && "/Users/jeevan/Documents/the_new_mission/"25_pattern
jeevan@Jeevans-MacBook-Air the_new_mission % cd "/Users/jeevan/Documents/the_new_mission/" && g++ 25_pattern.cpp -o 25_pattern && "/Users/jeevan/Documents/the_new_mission/"2
5_pattern
 @  @  @  @  @
 @           @
 @           @
 @           @
 @           @
 @  @  @  @  @
jeevan@Jeevans-MacBook-Air the_new_mission %
```

Output: (screenshot)

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    Code

cd "/Users/jeevan/Documents/the_new_mission/" && g++ 25_pattern.cpp -o 25_pattern && "/Users/jeevan/Documents/the_new_mission/"25_pattern
jeevan@Jeevans-MacBook-Air the_new_mission % cd "/Users/jeevan/Documents/the_new_mission/" && g++ 25_pattern.cpp -o 25_pattern && "/Users/jeevan/Documents/the_new_mission/"2
5_pattern
 @  @  @  @  @
 @           @
 @           @
 @           @
 @           @
 @  @  @  @  @
jeevan@Jeevans-MacBook-Air the_new_mission %
```

Test Case: Any two (screenshot)

Conclusion:

This C++ program demonstrates the use of nested loops and conditional statements to generate a specific pattern using the '@' symbol. The pattern includes a line of '@' symbols, spaces, and additional '@' symbols forming a diagonal line and a staircase-like structure.

The program provides an example of how loops and conditional statements can be combined to create visually interesting patterns. While this specific pattern may not have a practical application, it serves as an illustrative example of controlling output in a structured manner using basic C++ constructs.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 15

Title: Write a C++ program to display the following pattern:

```
    1
   232
  34543
 4567654
  34543
   232
    1
```

Theory:

1. Nested Loops:

   - Two sets of nested `for` loops are used for the upper and lower parts of the pyramid.

2. Control Structures:

   - The program uses loops and conditional statements to control the iteration and printing of numbers.

3. Variable Manipulation:

   - The variable `num` is manipulated to control the sequence of numbers on each row.

Code:

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

int main()
{
    int n = 5, i, j, num = 1, gap;
    gap = n - 1;

    for ( j = 1 ; j < n ; j++ )
    {
        num = j;

        for ( i = 1 ; i <= gap ; i++ )
            cout << " ";

        gap --;

        for ( i = 1 ; i <= j ; i++ )
```

```
        {
            cout << num;
            num++;
        }


        num--;
        num--;


        for ( i = 1 ; i < j ; i++)
        {
            cout << num;
            num--;
        }


        cout << "\n";
    }


    for ( j = n-1 ; j >= 1 ; j-- )
    {
        num = j;


        for ( i = 1 ; i <= n-j ; i++ )
            cout << " ";


        for ( i = 1 ; i <= j ; i++ )
        {
            cout << num;
            num++;
        }


        num--;
        num--;


        for ( i = 1 ; i < j ; i++)
        {
            cout << num;
            num--;
        }


        cout << "\n";
    }
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    Code

 cd "/Users/jeevan/Documents/the_new_mission/" && g++ 26_pattern.cpp -o 26_pattern && "/Users/jeevan/Documents/the_new_mission/"26_pattern
jeevan@Jeevans-MacBook-Air the_new_mission % cd "/Users/jeevan/Documents/the_new_mission/" && g++ 26_pattern.cpp -o 26_pattern && "/Users/jeevan/Documents/the_new_mission/"2
6_pattern
    1
   232
  34543
 4567654
 4567654
  34543
   232
    1
jeevan@Jeevans-MacBook-Air the_new_mission %
```

Output: (screenshot)

Test Case: Any two (screenshot)

Conclusion:

This C++ program showcases the use of nested loops and variable manipulation to create a visually appealing pyramid pattern with a numeric sequence. The pattern consists of ascending and descending numbers on each row, forming a symmetrical structure.

While this specific pattern may not have a practical application, the program serves as an educational example of how loops and conditional statements can be used to control the structure of output in a C++ program. It also demonstrates the manipulation of variables to achieve a specific sequence in the pattern.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 16

Title: Write a program to creating an inventory management system for a small store. The system should use object-oriented principles in C++. Yourprogram should have the following features:

- Create a **Product** class that represents a product in the inventory. Each **Product** object should have the following attributes:
  - Product ID (an integer)
  - Product Name (a string)
  - Price (a floating-point number)
  - Quantity in stock (an integer)
- Implement a parameterized constructor for the **Product** class toinitialize the attributes when a new product is added to the inventory

Theory:

1. Class Definition:

- The `Product` class is defined with private attributes: `pro_id`, `pro_name`, `price`, and `quantity`.

- It includes a default constructor and a parameterized constructor to initialize the attributes.

2. Array of Objects:

- An array of `Product` objects (`p[a]`) is created to store details for multiple products.

3. Object Initialization and Display:

- The user is prompted to input details for each product using a loop.

- The details entered are used to create and initialize `Product` objects, and the details are then displayed.

4. Encapsulation:

- The use of private attributes in the `Product` class demonstrates encapsulation, where data is hidden and accessed through public methods (constructors in this case).

Code:

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

class Product
 {
    private:
      int pro_id;
      string pro_name;
      float price;
      int quantity;

    public:
      Product()
      {}
      Product(int id, string name, float mrp,int quant )
        {
          pro_id= id;
          pro_name = name;
          price = mrp;
          quantity= quant;

          cout <<endl<<"Product ID: "<<pro_id<< endl;
          cout <<"Product Name: "<<pro_name<< endl;
          cout <<"Price: "<<price<< endl;
          cout <<"Quantity: "<<quantity<< endl;
        }
 };

int main()
{
  int a, b, e;
  float d;
  string c;
  cout<< "Enter the no. of products: ";
  cin>> a;

  Product p[a];

  int s=1;

  for (int i = 0; i < a; i++)
  {
    cout<< "For Product "<< s++ << ": " <<endl;
```

```
    cout<<"Enter the Product ID: ";
    cin>>b;

    cout<<"Enter the Product Name: ";
    cin>> c;

    cout<<"Enter the Product Price: ";
    cin>> d;

    cout<<"Enter the Product Quantity: ";
    cin>> e;

    p[i] = Product (b, c, d, e) ;
  }

  return 0;
}
```

```
cd "/Users/jeevan/Documents/Start/" && g++ 1_inventry.cpp -o 1_inventry && "/Users/jeevan/Documents/Start/"1_inventry
jeevan@Jeevans-MacBook-Air the_new_mission % cd "/Users/jeevan/Documents/Start/" && g++ 1_inventry.cpp -o 1_inventry && "/Users/jeevan/Documents/Start/"1_inventry
Enter the no. of products: 3
For Product 1:
Enter the Product ID: 1
Enter the Product Name: asds
Enter the Product Price: 12
Enter the Product Quantity: 2

Product ID: 1
Product Name: asds
Price: 12
Quantity: 2
For Product 2:
Enter the Product ID:

2
Enter the Product Name: dfsg
Enter the Product Price:
50
Enter the Product Quantity: 5

Product ID: 2
Product Name: dfsg
Price: 50
Quantity: 5
For Product 3:
Enter the Product ID: 3
Enter the Product Name: gjghf
Enter the Product Price: 45
Enter the Product Quantity: 3

Product ID: 3
Product Name: gjghf
Price: 45
Quantity: 3
jeevan@Jeevans-MacBook-Air Start %
```

Output: (screenshot)

Test Case: Any two (screenshot)

```
cd "/Users/jeevan/Documents/Start/" && g++ 1_inventry.cpp -o 1_inventry && "/Users/jeevan/Documents/Start/"1_inventry
jeevan@Jeevans-MacBook-Air the_new_mission % cd "/Users/jeevan/Documents/Start/" && g++ 1_inventry.cpp -o 1_inventry && "/Users/jeevan/Documents/Start/"1_inventry
Enter the no. of products: 1
For Product 1:
Enter the Product ID: 1
Enter the Product Name: saddf
Enter the Product Price: 13
Enter the Product Quantity: 5

Product ID: 1
Product Name: saddf
Price: 13
Quantity: 5
jeevan@Jeevans-MacBook-Air Start %
```

```
cd "/Users/jeevan/Documents/Start/" && g++ 1_inventry.cpp -o 1_inventry && "/Users/jeevan/Documents/Start/"1_inventry
jeevan@Jeevans-MacBook-Air the_new_mission % cd "/Users/jeevan/Documents/Start/" && g++ 1_inventry.cpp -o 1_inventry && "/Users/jeevan/Documents/Start/"1_inventry
Enter the no. of products: 2
For Product 1:
Enter the Product ID: 1
Enter the Product Name: dsag
Enter the Product Price: 13
Enter the Product Quantity: 5

Product ID: 1
Product Name: dsag
Price: 13
Quantity: 5
For Product 2:
Enter the Product ID: 2
Enter the Product Name: adhgf
Enter the Product Price: 20
Enter the Product Quantity: 1

Product ID: 2
Product Name: adhgf
Price: 20
Quantity: 1
jeevan@Jeevans-MacBook-Air Start %
```

Conclusion:

This C++ program illustrates the concept of classes and objects. The `Product` class serves as a blueprint for creating objects representing individual products. The program allows the user to input details for a specified number of products, creating an array of objects. This example demonstrates the encapsulation principle by keeping the data members private and accessing them through public methods.

Additionally, the program emphasizes the use of constructors (both default and parameterized) to initialize object attributes during object creation. It provides a practical example of how object-oriented programming concepts can be applied to model and represent real-world entities in a program.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 17

Title: Write a program to manage student records. Create a class Student with attributes such as name, roll number, and marks. Implement methods for displaying student details, adding new students, and calculating the average marks of all students in the record system.

Theory:

1. Class Definition:
   - The `Students` class is defined with private attributes: `name`, `roll_number`, and `marks`.
   - It includes a default constructor and a parameterized constructor to initialize the attributes.

2. Array of Objects:
   - An array of `Students` objects (`p[a]`) is created to store details for multiple students.

3. Object Initialization and Input:
   - The user is prompted to input details for each student using a loop.
   - Validation is applied to ensure that the marks entered are not greater than 100.

4. Encapsulation:
   - The use of private attributes in the `Students` class demonstrates encapsulation.

5. Calculating Total and Average Marks:
   - The program calculates the total marks and average marks based on user input.

Code:

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

class Students
{

    private:
      string name;
      int roll_number;
      int marks;

    public:
      Students()
       {}
      Students(string name, int roll, int mark)
       {
            name = name;
            roll_number=roll;
            marks=mark;

       }
};

int main()
{
    int a;
    cout<< "Enter the number of Students: ";
    cin>> a;

    int c,d;
    string b;
    int sum=0;

    Students p[a];

    for (int i = 0; i < a; i++)
    {
        cin.ignore();
        cout<<"Name: "<<endl;
        getline(cin, b);

        cout<< "Roll no:. "<<endl;
        cin>> c;

        here:
        cout<<"Marks: "<<endl;
```

```
        cin>> d;

        if (d> 100)
        {
            goto here;
        }

        sum += d;
        p[i] = Students (b, c, d) ;
    }

  cout<< "Total: "<<sum<<endl;
  cout<< "average: "<<sum/a<<endl;


  return 0;
}
```

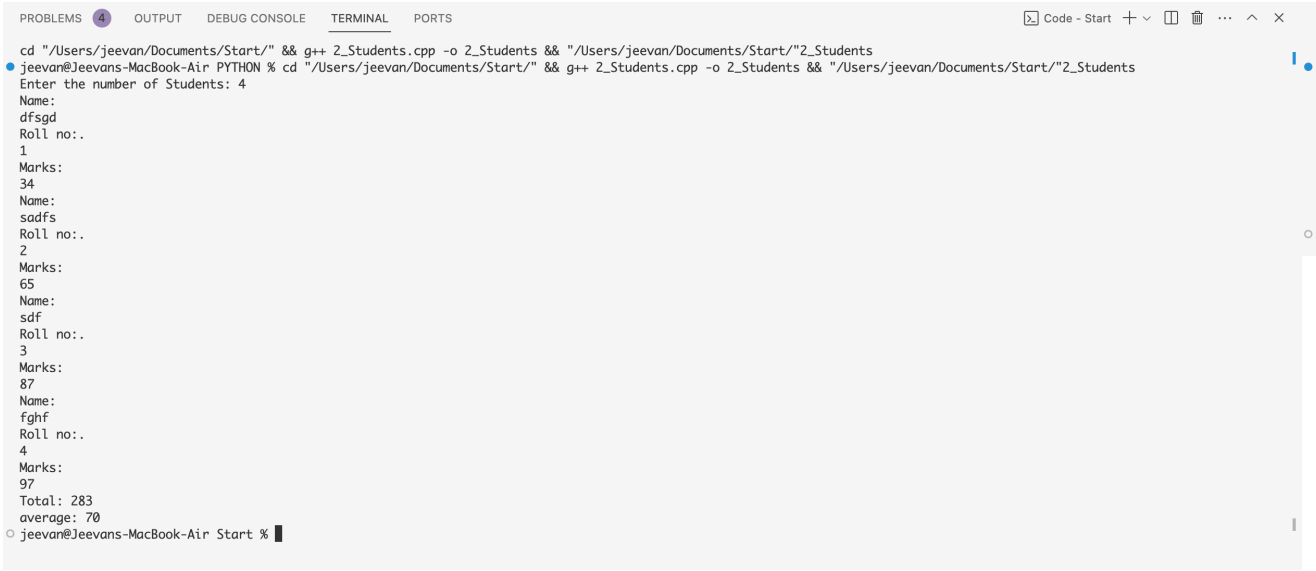
Output: (screenshot)

Test Case: Any two (screenshot)

Conclusion:

This C++ program illustrates the concept of classes and objects to represent student details. The `Students` class serves as a blueprint for creating objects representing individual students. The program allows the user to input details for a specified number of students, creating an array of objects.

Validation is applied to ensure that the entered marks are not greater than 100. The program then calculates and displays the total marks and average marks of the entered students. This example



demonstrates the encapsulation principle and the use of object-oriented programming concepts to model and represent real-world entities in a program.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 18

Title: Write a program that implements a basic calculator. Use a class Calculator with methods to perform addition, subtraction, multiplication, and division of two numbers. The program should allow the user to input two numbers and select an operation to perform.

Theory:

1. Class Definition (`calculator`):

- The `calculator` class includes private member variables `num1`, `num2`, `sum`, `sub`, `mult`, and `div` to store numbers and results.

- Public methods include `input` for user input and methods for addition, subtraction, multiplication, and division.

2. Input Method (`input`):

- The `input` method prompts the user to enter two numbers (`num1` and `num2`).

3. Arithmetic Operation Methods:

- Methods like `addition`, `subtraction`, `multiplication`, and `division` perform the respective operations and return the result.

4. Main Function:

- An instance of the `calculator` class (`calculate`) is created in the `main` function.

- The `input` method is called to get user input for two numbers.

- The user is prompted to enter the choice of operation (`+, -, *, /`).

- The appropriate operation method is called based on user input, and the result is displayed.

Code:

```
/*Create a C++ program that implements a basic calculator. Use a class
Calculator with methods to perform addition, subtraction, multiplication,
and division of two numbers. The program should allow the user to input
two numbers and select an operation to perform*/

#include <iostream>
```

```
using namespace std;

class calculator
{
   private:
    double num1, num2, sum,sub, mult, div;

   public:
    void input()
     {
       cout<<"Enter the first number: ";
       cin>>num1;
       cout<<"Enter the second number: ";
       cin>> num2;
     }
    double addition()
     {
        sum=num1 +num2;
        return sum;
     }
    double subtraction()
     {
        sub=num1-num2;
        return sub;
     }
    double multiplication()
     {
        mult = num1*num2;
        return mult;
     }
    double division()
     {
        div= num1/num2;
        return div;
     }
};

int main()
{
    calculator calculate;
    calculate.input();

    char a;
    cout<<"Enter the choice of operation(+, -, *, /): ";
    cin>>a;

    if (a == '+')
    {
        cout<< calculate.addition();
```

```
    }
    else if (a == '-')
    {
        cout<< calculate.subtraction();
    }
    else if (a =='*')
    {
        cout<<calculate.multiplication();
    }
    else
    {
        cout<<calculate.division();
    }

    return 0;
}
```

Output: (screenshot)

Test Case: Any two (screenshot)

Conclusion:

This C++ program demonstrates the use of a class to create a simple calculator. It allows the user to input two numbers and select an operation to perform (addition, subtraction, multiplication, or division). The program uses object-oriented principles to organize code and encapsulate functionality within the `calculator` class. It also showcases the use of conditional statements to determine the chosen operation and display the result accordingly.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 19

Title: Write a program to simulate a simple online shop. Create a class Product with attributes like name, price, and quantity in stock. Implement methods for adding products to the shopping cart, calculating the total cost, and displaying the contents of the cart.

Theory:

1. Class Definition (`Product`):

- The `Product` class includes private member variables `pro_name`, `price`, and `quantity` to represent product details.
- The class has a default constructor and a parameterized constructor to initialize the attributes.
- The `Data` method is a public member function to display product details.

2. Object Array and Input:

- An array of `Product` objects (`p[a]`) is created to store details for multiple products.
- The user is prompted to input details for each product using a loop.

3. Calculating Total Invoice:

- The program calculates the total invoice amount by multiplying the price and quantity for each product and adding the results.

4. Displaying Product Details:

- Another loop is used to display the product details using the `Data` method of each `Product` object.

Code:

```
#include <iostream>
```

```
using namespace std;

class Product
 {
    private:
      string pro_name;
      double price;
      int quantity;

    public:
      Product()
      {}
      Product(string name, float mrp,int quant )
        {
          pro_name = name;
          price = mrp;
          quantity= quant;
        }
       void Data()
       {
        cout <<"Product Name: "<<pro_name<< endl;
        cout <<"Price: "<<price<< endl;
        cout <<"Quantity: "<<quantity<< endl;
       }
 };

int main()
{
  int a, e;
  double d;
  string c;
  cout<< "Enter the no. of products: ";
  cin>> a;

  Product p[a];

  int s=1;
  double sum=0;

  for (int i = 0; i < a; i++)
  {
    cout<< "For Product "<< s++ << ": " <<endl;

    cout<<"Enter the Product Name: ";
    cin>> c;

    cout<<"Enter the Product Price: ";
    cin>> d;
```

```
    cout<<"Enter the Product Quantity: ";
    cin>> e;

    sum+=(d*e);

    p[i] = Product(c,d,e);
  }

  for (int x = 0; x < a; x++)
  {
    p[x].Data();
  }

  cout<<endl<< "Total Invoice: "<<sum<<endl;

  return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                   Code - Start

cd "/Users/jeevan/Documents/Start/" && g++ 4_online_shop.cpp -o 4_online_shop && "/Users/jeevan/Documents/Start/"4_online_shop
jeevan@Jeevans-MacBook-Air the_new_mission % cd "/Users/jeevan/Documents/Start/" && g++ 4_online_shop.cpp -o 4_online_shop && "/Users/jeevan/Documents/Start/"4_online_shop
Enter the no. of products: 2
For Product 1:
Enter the Product Name: fda
Enter the Product Price: 12
Enter the Product Quantity: 3
For Product 2:
Enter the Product Name: af
Enter the Product Price: 2
Enter the Product Quantity: 45
Product Name: fda
Price: 12
Quantity: 3
Product Name: af
Price: 2
Quantity: 45

Total Invoice: 126
jeevan@Jeevans-MacBook-Air Start %
```

Output: (screenshot)

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                   Code - Start

cd "/Users/jeevan/Documents/Start/" && g++ 4_online_shop.cpp -o 4_online_shop && "/Users/jeevan/Documents/Start/"4_online_shop
jeevan@Jeevans-MacBook-Air the_new_mission % cd "/Users/jeevan/Documents/Start/" && g++ 4_online_shop.cpp -o 4_online_shop && "/Users/jeevan/Documents/Start/"4_online_shop
Enter the no. of products: 1
For Product 1:
Enter the Product Name: asf
Enter the Product Price: 70
Enter the Product Quantity: 5
Product Name: asf
Price: 70
Quantity: 5

Total Invoice: 350
jeevan@Jeevans-MacBook-Air Start %
```

Test Case: Any two (screenshot)

Conclusion:

This C++ program demonstrates the use of a class to create a simple inventory system. The `Product` class encapsulates product details, and an array of objects is used to store information for multiple products. The program allows the user to input details, calculates the total invoice amount, and displays product details along with the total invoice. It illustrates the principles of object-oriented programming, encapsulation, and the organization of code into meaningful classes and methods.

Name of Student: Jeevan Naidu

Roll Number: 15

Experiment No: 20

Title: Write a program to manage student grades for a classroom. Create a class Student with attributes for student name and an array to store grades. Implement methods for adding grades, calculating the average grade, and displaying the student's name and grades. Use constructors and destructors to initialize and release resources.

Theory:

1. Constructor Issue:

- In the `Students` class parameterized constructor, the member variables `name`, `roll_number`, and `marks` are assigned the same names as the constructor parameters. This can lead to confusion and incorrect results. The member initializer list or `this` pointer should be used to distinguish between parameters and member variables.

2. Destructor (`~Students`):

- The code includes a destructor for the `Students` class, but it does not perform any specific cleanup. In this case, it's unnecessary, and the default destructor provided by the compiler would suffice.

3. Grade Calculation:

- The grade calculation (`if` statements) in the `main` function uses the variable `d` outside the loop, which is not appropriate. It should be inside the loop and calculated based on the marks entered for each student.

4. Grade Criteria:

- The grade criteria seem to be hard-coded, and the logic might not be suitable for a typical grading system. It assigns grades based on specific numeric values without considering the overall distribution of marks.

Code:

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

class Students
{

    private:
```

```
      string name;
      int roll_number;
      int marks;

    public:
      Students()
       {}
      Students(string name, int roll, int mark)
       {
            name = name;
            roll_number=roll;
            marks=mark;

       }
      ~Students()
      {}
};

int main()
{
    int a;
    cout<< "Enter the number of Students: ";
    cin>> a;

    int c,d;
    string b;
    int sum=0;

    Students p[a];

    for (int i = 0; i < a; i++)
    {
        cin.ignore();
        cout<<"Name: "<<endl;
        getline(cin, b);

        cout<< "Roll no:. "<<endl;
        cin>> c;

        here:
        cout<<"Marks: "<<endl;
        cin>> d;

        if (d> 10)
        {
            goto here;
        }

        sum += d;
        p[i] = Students (b, c, d) ;
    }

  cout<< "Total: "<<sum<<endl;
  cout<< "average: "<<sum/a<<endl;
  if (d>7)
        {
            cout<<"Grade: A";
        }
```

```
        else if (d >5 && d<7)
        {
            cout<<"Grade: B";
        }
        else if (d>3 && d<5)
        {
            cout<<"Grade: c";
        }
        else
        {
            cout<<"Grade: Fail !";
        }

  return 0;
}
```

```
PROBLEMS 4  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                   Code - Start

cd "/Users/jeevan/Documents/Start/" && g++ 5_student.cpp -o 5_student && "/Users/jeevan/Documents/Start/"5_student
jeevan@Jeevans-MacBook-Air PYTHON % cd "/Users/jeevan/Documents/Start/" && g++ 5_student.cpp -o 5_student && "/Users/jeevan/Documents/Start/"5_student
Enter the number of Students: 5
Name:
a
Roll no:.
1
Marks:
9
Name:
b
Roll no:.
2
Marks:
5
Name:
c
Roll no:.
3
Marks:
5
Name:
d
Roll no:.
4
Marks:
1
Name:
e
Roll no:.
5
Marks:
0
Total: 20
average: 4
Grade: Fail !
jeevan@Jeevans-MacBook-Air Start %
```

Output: (screenshot)

```
PROBLEMS 4  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                   Code - Start

cd "/Users/jeevan/Documents/Start/" && g++ 5_student.cpp -o 5_student && "/Users/jeevan/Documents/Start/"5_student
jeevan@Jeevans-MacBook-Air PYTHON % cd "/Users/jeevan/Documents/Start/" && g++ 5_student.cpp -o 5_student && "/Users/jeevan/Documents/Start/"5_student
Enter the number of Students: 2
Name:
a
Roll no:.
1
Marks:
9
Name:
b
Roll no:.
2
Marks:
8
Total: 17
average: 8
Grade: A
jeevan@Jeevans-MacBook-Air Start %
```

Test Case: Any two (screenshot)

Conclusion:

The program aims to capture student details, calculate total marks, average marks, and assign grades. However, there are issues related to variable names, destructor usage, and the calculation of grades. It's essential to ensure that the member variables and parameters in the class constructor have distinct names. Additionally, the grading logic should be reviewed to align with standard grading criteria.

After addressing these issues, the program should work more accurately in capturing student information and providing meaningful grades based on a more realistic grading system.