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

C++ PROGRAMMING LAB

Prepared by:

Name of Student: PIYUSH KUMAR SINGH

Roll No: 43

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:- PIYUSH KUMAR SINGH

Roll Number:- 43

Experiment No:- 1

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

Theory:-

- **The program calculates roots of a quadratic equation using the quadratic formula: x = (-b ± sqrt(b^2 - 4ac)) / (2a).**
- **It uses if-else statements to handle real, equal, or complex roots based on the discriminant (b^2 - 4ac).**

Code:-

```
//  implement a program to find the roots of a quadratic equation.

#include <iostream>
#include <cmath>
```

```cpp
using namespace std;
void findRootsOfQuadraticEquation(double a, double b, double c) {
    double discriminant = b * b - 4 * a * c;

    if (discriminant > 0) {
        double root1 = (-b + sqrt(discriminant)) / (2 * a);
        double root2 = (-b - sqrt(discriminant)) / (2 * a);
        cout << "Root 1: " << root1 << endl;
        cout << "Root 2: " << root2 << endl;
    } else if (discriminant == 0) {
        double root = -b / (2 * a);
        cout << "Root: " << root << endl;
    } else {
        double realPart = -b / (2 * a);
        double imaginaryPart = sqrt(-discriminant) / (2 * a);
        cout<< "Root 1: " << realPart << " + " << imaginaryPart <<
"i" << endl;
        cout << "Root 2: " << realPart << " - " << imaginaryPart <<
"i" << endl;
    }
}

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

    cout << "Enter coefficient a: ";
    cin >> a;

    cout << "Enter coefficient b: ";
    cin >> b;

    cout << "Enter coefficient c: ";
    cin >> c;

    if (a == 0) {
        double x=-c/b;
        cout<<"Linear equation.\nroot ="<<x<<"\n";
    }

    cout << "Quadratic equation: " << a << "x^2 + " << b << "x + "
<< c << " = 0" << endl;

    findRootsOfQuadraticEquation(a, b, c);

    return 0;
}
```

Output:-

```
Enter coefficient a: 2
Enter coefficient b: 3
Enter coefficient c: 4
Quadratic equation: 2x^2 + 3x + 4 = 0
Root 1: -0.75 + 1.19896i
Root 2: -0.75 - 1.19896i
```

Caption

Test case:-

```
Enter coefficient a: 10
Enter coefficient b: 20
Enter coefficient c: 30
Quadratic equation: 10x^2 + 20x + 30 = 0
Root 1: -1 + 1.41421i
Root 2: -1 - 1.41421i
```

Caption

Conclusion:-

THE PROGRAM USES THE QUADRATIC FORMULA TO FIND THE ROOTS OF A QUADRATIC EQUATION, HANDLING BOTH REAL AND COMPLEX ROOTS BASED ON THE DISCRIMINANT. USERS CAN INPUT THE COEFFICIENTS, AND THE PROGRAM OUTPUTS THE CORRESPONDING ROOTS OF THE QUADRATIC EQUATION.

Experiment No: 2

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

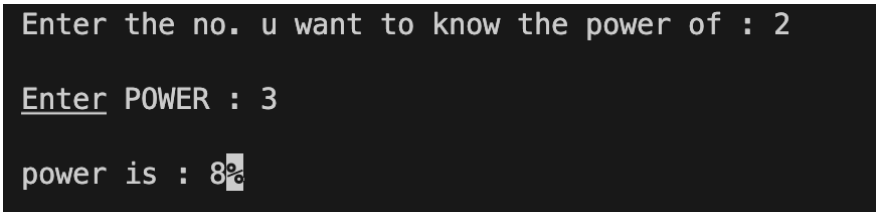
Theory:-

- **Employ a loop to repeatedly multiply the base number by itself, simulating exponentiation.**
- **Continuously update the result in each iteration to capture the increasing power.**

Code:-

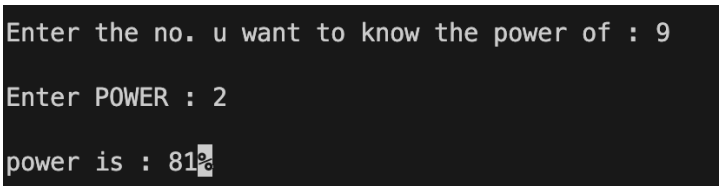
```
//Create a program to calculate the power of a number using a loop.

#include <iostream>
using namespace std;

int main()
{
    int n ,p;
    long ans = 1;

    cout<<"\nEnter the no. u want to know the power of : ";
    cin>>n;

    cout<<"\nEnter POWER : ";
    cin>>p;

    for (int i = 1; i <= p; i++)
    {
        ans=ans * n;
    }
    cout<<"\npower is : "<<ans;


return 0;

}
```

Output:-



Caption

Test Case:-



Caption

Conclusion:-

- **The program demonstrates an iterative approach for calculating the power of a number.**
- **It delivers an effective solution through repetitive multiplication in a loop.**

Experiment No:3

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

Theory:-

- **Iterate through the string, comparing characters from both ends to check for symmetry.**
- **Determine if the string reads the same forward and backward.**

Code:-

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

int main() {
    string input;
    int choice;

    do {
        cout << "Please enter a string: ";
        cin >> input;

        int left = 0;
        int right = input.length() - 1;
        bool isPalindrome = true;

        while (left < right)
        {
            if (input[left] != input[right])
            {
                isPalindrome = false;
                break;
            }
            left++;
            right--;
        }

        if (isPalindrome==true)
        {
            cout << "It is a palindrome";
```

```
        } else
        {
            cout << "Not a palindrome";
        }

        cout << endl << "Check for another? " << endl;

        do {
            cout << "Press 1 for yes, 2 for No" << endl;
            cin >> choice;

            if (choice != 2 && choice != 1)
            {
                cout << "Please enter a valid response" << endl;
            }
        } while (choice != 2 && choice != 1);

        if (choice == 2)
        {
            cout << "Thank you for using this program, have a great
day!! ";
            break;
        }
    } while (true);

    return 0;
}
```

OUTPUT:-

```
Please enter a string: das
Not a palindrome
Check for another?
Press 1 for yes, 2 for No
2
Thank you for using this program, have a great day!!
```

Caption

Test case:-

```
It is a palindrome
Check for another?
Press 1 for yes, 2 for No
1
Please enter a string: madam
It is a palindrome
Check for another?
Press 1 for yes, 2 for No
```

Caption

Conclusion:-

f the string and its reverse are the same, it's a palindrome; otherwise, it's not.

Experiment No:4

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:-

- **The program emulates an ATM using a switch statement.**
- **Users choose options (check balance, deposit, or withdraw) to perform corresponding actions on their account.**

Code:-

```
/*          Implement a C++ program that simulates a simple ATM
machine,
 allowing users to check their balance, deposit,or withdraw money
using a switch statement. */

#include <iostream>
#include <unistd.h>
using namespace std;

int main()
{
    int n,a, r,w, d, t,choice;

    start:

    cout<<"\n\nENTER YOUR ACCOUNT NO.";
    cin>>n;
    if (n<=1000 || n>=9999)
    {
        cout<<" INVALID NO.! ENTER 4 DIGIT NO.";
        goto start;
        return 0;
    }

    srand((int) time(NULL));
    r = rand()%10000;
    sleep(2);
```

```
    menu:

    cout<<"\n\nWHAT U WANT TO DO ?\n";
    cout<<"1. Withdrow Money\n";
    cout<<"2. Check Money\n";
    cout<<"3. Deposit Blance\n";
    cout<<"4. Transfer Money\n";
    cout<<"5. Exit\n";
    cin>>choice;


    switch (choice)
    {
    case 1:

        cout<<"HOW MUCH MONEY U WANT TO WITHDROW : ";
        cin>>w;
        sleep(2);

        cout<<"MONEY WITHDROW SUCCESSFULLY \n";
        r = r - w;
        cout<<"\nNOW YOU HAVE "<< r<<" IN YOUR BANK ACCOUNT";
        break;


    case 2:

        cout<<"YOUR ACCOUNT HAVE : "<<"$"<<r;
        break;


    case 3:

        cout<<"HOW MUCH MONEY U WANT TO DEPOSIT : ";
        cin>>d;
        r+=d;
        cout<<"\n\nMONEY DEPOSIT SUCCESSFULLY\n ";
        cout<<"NOW YOU HAVE "<< r<<" IN YOUR BANK ACCOUNT";
        break;


    case 4:

        cout<<"YOUR ACCOUNT HAVE : "<<"$"<<d<<"\n";

        cout<<"\nENTER  THE ACCOUNT NUMBER U WANT TRASFER: ";
        cin>>a;
        sleep(1);
```

```
        cout<<"\nHOW MUCH MONEY U WANT TO TRANSFER : ";
        cin>>t;
        sleep(1);

        r-=t;

        cout<<"MONEY TRANSFER SUCCESSFULLY \n";
        cout<<"\nNOW YOU HAVE "<< r<<" IN YOUR BANK ACCOUNT ";
        break;

    case 5:
        cout<<"Thank you for choosing us :) \n";
        return 0;
    default:
     cout<<"INVALID CHOICE";

     break;
    }
    sleep(1);
    goto menu;

  return 0;
}
```

Output:-

```
ENTER YOUR ACCOUNT NO.1234


WHAT U WANT TO DO ?
1. Withdrow Money
2. Check Money
3. Deposit Blance
4. Transfer Money
5. Exit
2
YOUR ACCOUNT HAVE : $1613
```

Caption

```
WHAT U WANT TO DO ?
1. Withdrow Money
2. Check Money
3. Deposit Blance
4. Transfer Money
5. Exit
4
YOUR ACCOUNT HAVE : $-2029386640

ENTER  THE ACCOUNT NUMBER U WANT TRASFER: 5678

HOW MUCH MONEY U WANT TO TRANSFER : 613
MONEY TRANSFER SUCCESSFULLY

NOW YOU HAVE 1000 IN YOUR BANK ACCOUNT
```

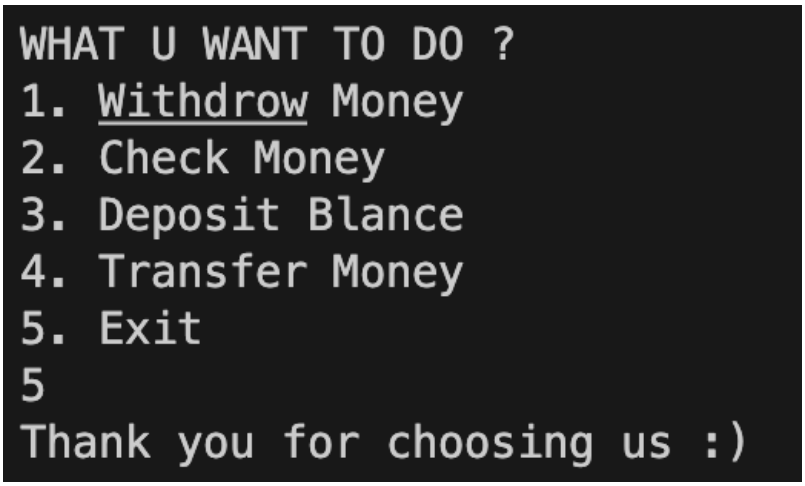
Caption

```
WHAT U WANT TO DO ?
1. Withdrow Money
2. Check Money
3. Deposit Blance
4. Transfer Money
5. Exit
3
HOW MUCH MONEY U WANT TO DEPOSIT : 1000


MONEY DEPOSIT SUCCESSFULLY
 NOW YOU HAVE 2000 IN YOUR BANK ACCOUNT
```

Caption

```
WHAT U WANT TO DO ?
1. Withdrow Money
2. Check Money
3. Deposit Blance
4. Transfer Money
5. Exit
1
HOW MUCH MONEY U WANT TO WITHDROW : 500
MONEY WITHDROW SUCCESSFULLY

NOW YOU HAVE 1500 IN YOUR BANK ACCOUNT
```

Caption



Caption

Confusion:-

- **Using the switch statement, the program facilitates diverse user interactions by executing specific actions based on the chosen option.**
- **This modular approach enhances code readability and user experience in the simulated ATM environment.**

Experiment No:5

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

Theory:-

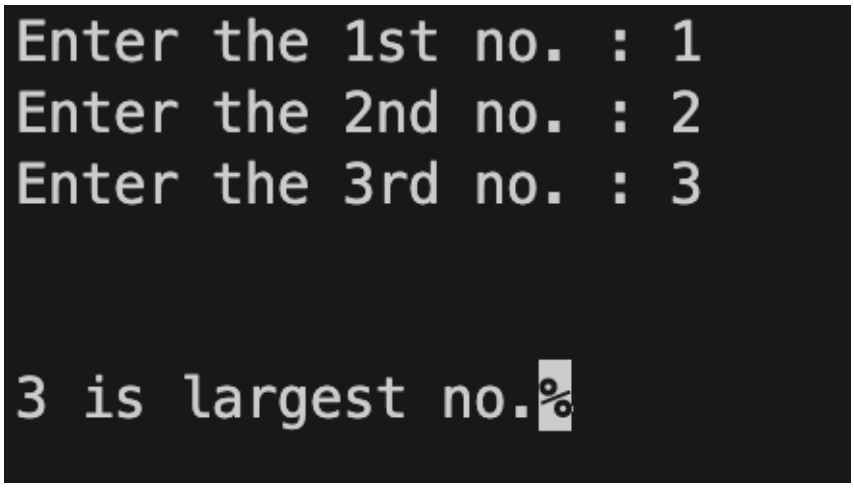
- **The program determines the largest number among three using nested if-else statements.**
- **It compares the numbers stepwise, checking conditions to identify and output the largest number.**

Code:-

```
//Implement a program that finds the largest among three numbers
using nested if-else statements.

#include <iostream>
using namespace std;

int main()
{
    int num1, num2, num3;

    cout<<"\n\nEnter the 1st no. : ";
    cin>>num1;

    cout<<"Enter the 2nd no. : ";
    cin>>num2;

    cout<<"Enter the 3rd no. : ";
    cin>>num3;

    if (num1>num2)
    {
        if (num1>num3)
        {
            cout<<"\n\n"<<num1<<"\n\n is the largest no.";
        }
```

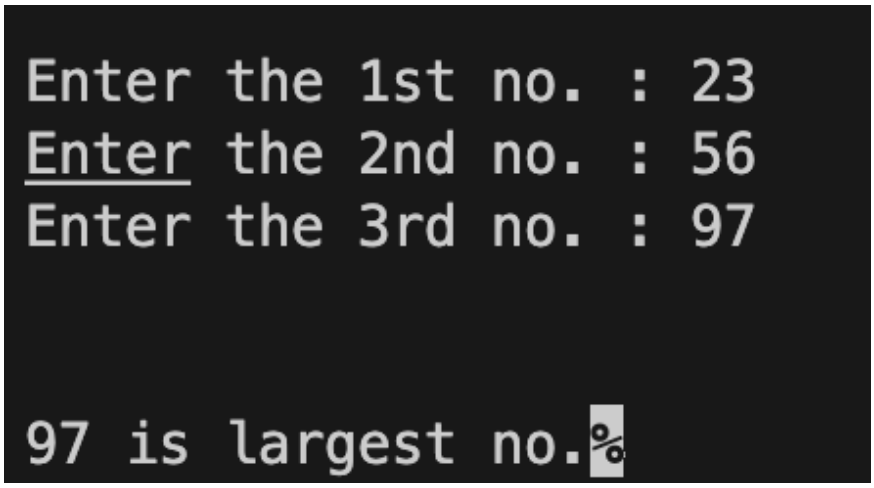
```
        else
        {
            cout<<"\n\n"<<num3<<" is the largest no.";

        }
    }

    else
    {
        if (num2>num1)
        {
            if (num2>num3)
            {
                cout<<"\n\n"<<num2<<" is largest no.";
            }
            else
            {
                cout<<"\n\n"<<num3<<" is largest no.";
            }
        }
    }

return 0;
}
```

Output:-



Caption

Test case:-



Caption

Conclusion:-

- **The nested if-else construct facilitates a systematic examination of numerical magnitudes, leading to the accurate identification of the largest among the three input values.**

- **This method enhances code clarity and ensures robust decision-making in determining the maximum number.**

Experiment No:6

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

Theory:-

- **The program evaluates a student's grade based on marks in 5 subjects using an if-else-if ladder.**
- **It compares the total marks to predefined grade boundaries to assign the corresponding grade.**

Code:-

```
//Implement a program that determines the grade of a student based
on their marks of 5 subject

#include <iostream>
using namespace std;

int main()
{
    int m,p,c,e,ave,sum;
    string name;

    cout<<"\n\nSTUDENT NAME : ";
    cin>>name;

    cout<<"\nMARKS OF MATH : ";
    cin>>m;

    cout<<"\nMARKS OF PHYSICS : ";
    cin>>p;

    cout<<"\nMARKS OF CHEMISTRY  : ";
    cin>>c;
```

```
    cout<<"\nMARKS OF ENGLISH : ";
    cin>>e;

   sum =m+p+c+e;
   ave=sum/4;

    cout<<"\nYOUR TOTAL AVERAGE MARKS IS : "<<ave;

    if (ave>=95 && ave<=100)
    {
        cout<<"\n Grade : 'A1' ";
    }
    else if (ave>=90 && ave<=95)
    {
        cout<<"\n Grade : 'A2' ";
    }

    else if (ave>=85 && ave<=90)
    {
        cout<<"\n Grade : 'B1' ";
    }

    else if (ave>=80 && ave<=85)
    {
       cout<<"\n Grade : 'B2' ";
    }

    else if (ave>=70 && ave<=80)
    {
        cout<<"\n Grade : 'C1' ";
    }

    else if (ave>=60 && ave<=69)
    {
        cout<<"\n Grade : 'C2' ";
    }

    else if (ave>=50 && ave<=59)
    {
        cout<<"\n Grade : 'C3' ";
    }

    else if (ave>=40 && ave<=49)
    {
        cout<<"\n Grade : 'D' ";
    }

    else if (ave>=35 && ave<=39)
    {
```

```
        cout<<"\n Grade : 'E' ";
    }

    else
    {
        cout<<"\n AAP APNE GHAR JA RAHE HO :)";
    }

return 0;
}
```

Output:-

```
STUDENT NAME : piyush

MARKS OF MATH : 99

MARKS OF PHYSICS : 99

MARKS OF CHEMISTRY  : 99

MARKS OF ENGLISH : 99

YOUR TOTAL AVERAGE MARKS IS : 99
 Grade : 'A1' %
```

Caption

Test case:-

```
STUDENT NAME : romil

MARKS OF MATH : 99

MARKS OF PHYSICS : 98

MARKS OF CHEMISTRY  : 78

MARKS OF ENGLISH : 67

YOUR TOTAL AVERAGE MARKS IS : 85
 Grade : 'B1' %
```

Caption

Conclusion:-

The if-else-if ladder efficiently categorizes students into grades according to their total marks, providing a straightforward method for assessing academic performance.

Experiment No:7

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

Theory:-

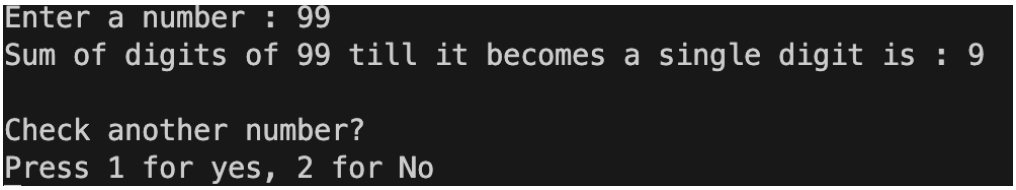
- **The program employs an iterative process to sum the digits of a number until it converges into a single-digit result.**
- **This involves repeatedly extracting and adding individual digits until a compact, single-digit sum is achieved.**

Code:-

```
// Write a C++ program to find the sum of digits of a number until
it becomes a single-digit number.

#include <iostream>
using namespace std;

int main()
{
    int n, choice;
    bool doagain = true;
    do
    {
        cout << "Enter a number : ";
        cin >> n;

        cout << "Sum of digits of " << n;
        int sum = 0;

        while (n > 0 || sum > 9)
        {
            if (n == 0)
            {
                n = sum;
```

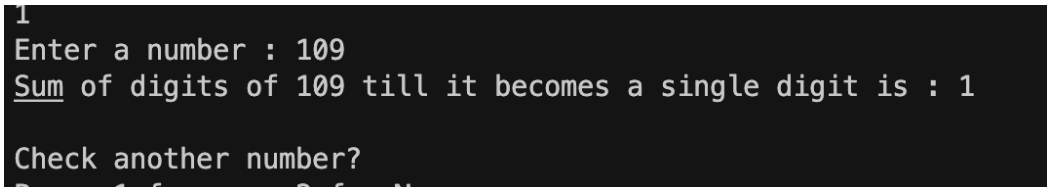
```
                sum = 0;
            }
            sum = sum + n % 10;
            n = n / 10;
        }
        cout << " till it becomes a single digit is : " << sum <<
endl;

        cout << endl<< "Check another number? " << endl;
        do
        {
            cout << "Press 1 for yes, 2 for No" << endl;
            cin >> choice;

            if (choice != 2 && choice != 1)
            {
                cout << "Please enter valid response " << endl;
            }
        } while (choice != 2 && choice != 1);

        if (choice == 2)
        {
            doagain = false;
            cout << "Thank you for using this program, have a great
day!!🤝🤝 ";
        }
    } while (doagain == true);

    return 0;
}
```

OUTPUT:-



Caption

Test case :-



Caption

Conclusion:-Through this iterative mechanism, the program ensures a comprehensive and systematic approach to finding the sum of digits, offering a concise and reliable method to obtain a single-digit outcome from any given numerical input.

Experiment No:8

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

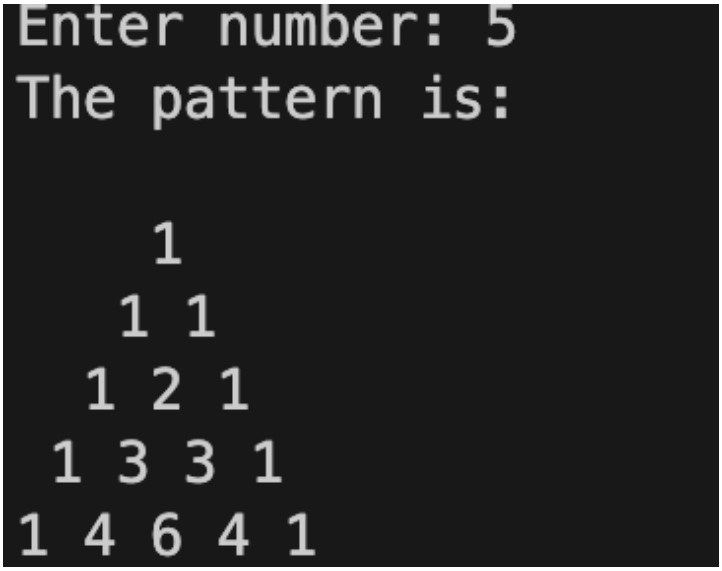
Theory:-

Code:-

```
// Implement a program to print a Pascal's triangle using nested
loops.

#include <iostream>
using namespace std;

// main function
int main() {

    // input a number from user
    int n;
    cout << "Enter number: ";
    cin >> n;

    while (n <= 0)
    {
        cout << "Invalid number" << endl;
        cout << "Enter number: " << endl;
        cin >> n;
    }

    // displaying the pattern
    cout << "The pattern is: " << endl << endl;

    for (int i = 1; i <= n; i++) {
        int num = 1;
        for (int j = 1; j <= n - i; j++) {
            cout << " ";
        }
        for (int k = 1; k <= i; k++) {
            cout << num << " ";
            num = num * (i - k) / k;
        }
```

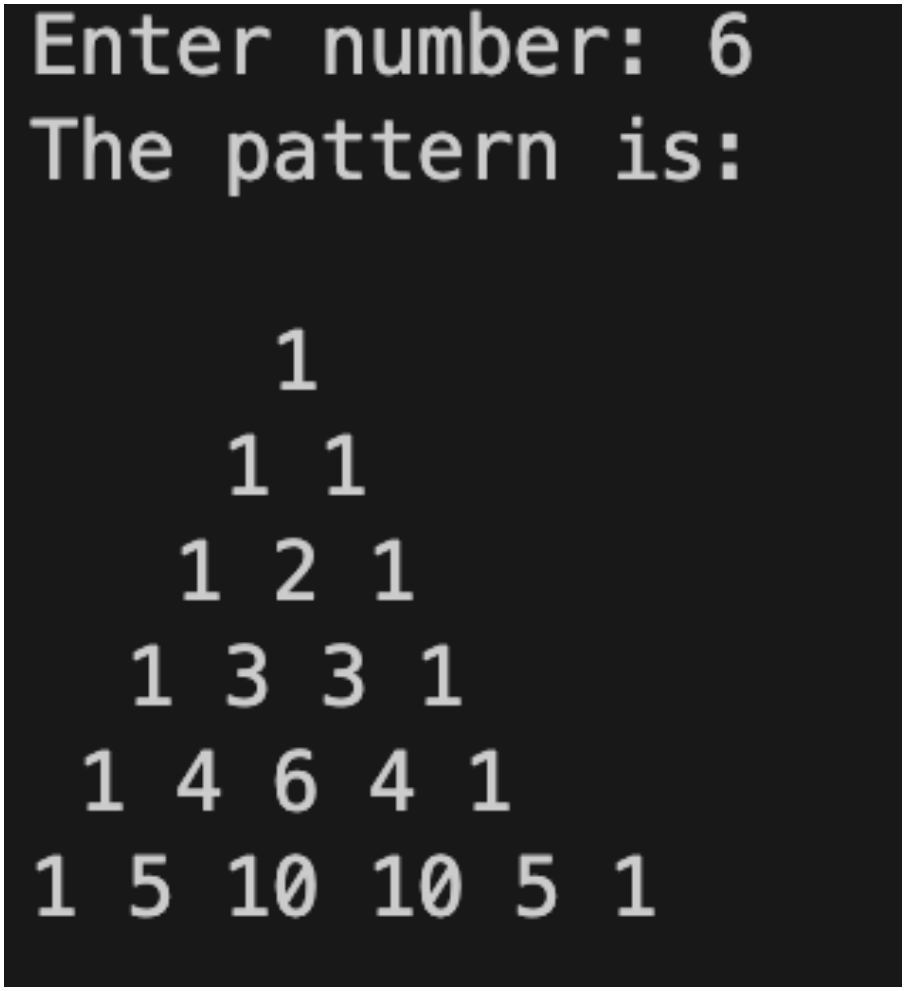
```
        cout << endl;
    }

    return 0;
}
```

Output:-



Caption

Test case:-



Caption

Conclusion:-

The nested loops efficiently construct Pascal's Triangle, revealing the pattern of binomial coefficients through systematic iteration and printing the triangular array.

Experiment No:9

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

Theory:-

- **The program employs nested loops to calculate the sum of the series 1/1! + 2/2! + 3/3! + ... + N/N!.**
- **It iteratively computes each term's value and accumulates the sum.**

Code:-

```
//  Calculate the sum of series 1/1! + 2/2! + 3/3! + ... + N/N!
using nested loops


#include <iostream>

using namespace std;

int main(){

    int n , p = 0 , q = 1 ;
    float sum = 0;


    cout<<"Number you want series till : \n";
    cin>>n;

    while(n>0)
    {
        for(int i = 1 ; i <= n ; i++)
        {
            p = i;
            q= q*i;

            if(i == n)
            {
```

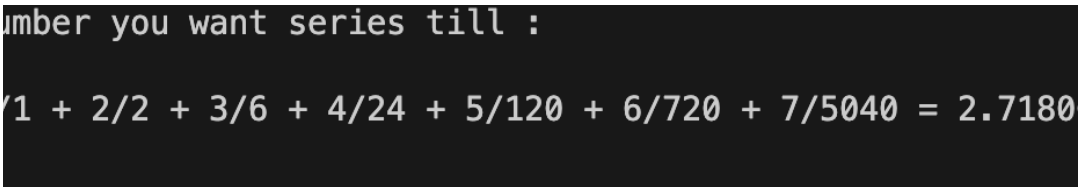
```
                cout<<p<<"/"<<q<<" = "<<sum+(float(p)/
float(q))<<"\n\n";
                return 0;

            }
            else
            {
                cout<<p<<"/"<<q<<" + ";
                sum += float(p)/float(q);
            }
}
    }

    cout<<"Invalid input please put positive number greater than
zero ..Thank you\n\n";

    return 0;
}
```

0utput:-

```
Number you want series till :
5
1/1 + 2/2 + 3/6 + 4/24 + 5/120 = 2.70833
```

Caption

Test case:-



Caption

Conclusion:-

The nested loops systematically evaluate and accumulate the series, providing an effective means to compute the sum of terms involving factorials in a structured manner.

Experiment No:10

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

Theory:-

- **The program creates an array of strings and arranges them in alphabetical order.**
- **It uses sorting algorithms to compare and reorder the strings within the array.**

Code:-

```
// Create an array of strings and display them in alphabetical
order.

#include <iostream>
#include <string>

using namespace std;

// function to sort an array using bubble sort
void sortArray(string arr[], int n)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n ; j++)
        {
            if (arr[j] > arr[j + 1])
            {
                string temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
```

```
    }
}

// main function
int main() {

    // input length and elements of the array from the user
    int lengthOfArray;
    cout << "Enter the length of the array: ";
    cin >> lengthOfArray;

    string array[lengthOfArray];

    for (int i = 0; i < lengthOfArray; i++) {
        cout << "Enter the " << i+1 << "th element of the array: ";
        cin >> array[i];
    }

    // calling the sorting function
    sortArray(array, lengthOfArray - 1);

    // displaying the sorted array
    cout << endl << "Names in alphabetical order:" << endl;
    for (int i = 0; i < lengthOfArray; ++i) {
        cout << array[i] << endl;
    }

    return 0;
}
```

Output:-

```
Enter the length of the array: 3
Enter the 1th element of the array: 12
Enter the 2th element of the array: 13
Enter the 3th element of the array: 14

Names in alphabetical order:
12
13
14
```

Caption

Test case:-

```
Enter the length of the array: 4
Enter the 1th element of the array: 1
Enter the 2th element of the array: 2
Enter the 3th element of the array: 3
Enter the 4th element of the array: 4

Names in alphabetical order:
1
2
3
4
```

Caption

Conclusion:-

By applying sorting algorithms, the program successfully organizes the array of strings alphabetically, ensuring a systematic presentation of the data in ascending order.

Experiment No:11

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

Theory:-

- **The program examines whether an array is sorted in ascending order.**
- **It iterates through the array, comparing each element with the next to verify the ascending order condition.**

Code:-

```
// Create a program that checks if an array is sorted in ascending order.

#include <iostream>

using namespace std;

int main()
{
    float temp;
    int n;
    bool check;

    cout<<"Enter the number of numbers you want to enter :  ";
    cin>>n;

    float arr[n];

    for(int i = 0 ; i < n ; i++)
    {
        cout<<"Enter "<<i+1<<" element";
        cin>>arr[i];
    }

    for(int i = 0 ; i < n-1 ; i++)
    {
        if(arr[i] > arr[i+1])
        {
```

```
            check = false;
        }
    }

    if(check == false)
    {
        cout<<"The array is not in ascending order.\nSorted array :
";
    }
    else{
        cout<<"Array is in ascending order ";
        return 0;
    }

    for(int i = 0 ; i < n ; i++)
    {
        for(int j = 0 ; j < n ; j++)
        {
            if(arr[i] < arr[j])
            {
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }

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

    return 0;
}
```

Output:-

```
Enter the number of numbers you want to enter :  4
Enter 1 element4
Enter 2 element3
Enter 3 element2
Enter 4 element1
The array is not in ascending order.
Sorted array : 1 2 3 4 %
```

Caption

Test case :-

```
Enter the number of numbers you want to enter :  3
Enter 1 element44
Enter 2 element43
Enter 3 element41
The array is not in ascending order.
Sorted array : 41 43 44 %
```

Caption

Conclusion:-

The program utilizes a systematic iteration process to determine if the array is sorted in ascending order, providing a reliable check for the arrangement of elements.

Experiment No:12

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

Theory:-

- **The program computes the sum of elements in each row of a matrix.**
- **It utilizes nested loops to iterate through rows and columns, accumulating the sum for each row.**

Code:-

```
// Calculate the sum of elements in each row of a matrix.

#include <iostream>
using namespace std;

int main() {
    int rows, col;
    cout << "Enter row and colum number: " << endl;
    cin >> rows >> col;

    //creating a 2d array and entering elements in it
    int arr[rows][col];
    for(int i = 0; i < rows; i++){
        for(int j = 0; j < col; j++){
            cout << "Enter value of arr["<< i <<"]["<< j << "]: ";
            cin >> arr[i][j];
        }
        cout << endl;
    }
    //adding elements of each row in the sum variable
    int sum=0;
    for(int i=0; i < rows; i++) {
        for (int j = 0; j < col; j++) {
            sum+=arr[i][j];
        }
        cout <<"Sum of row" <<i << "is:"<<sum << endl;
```

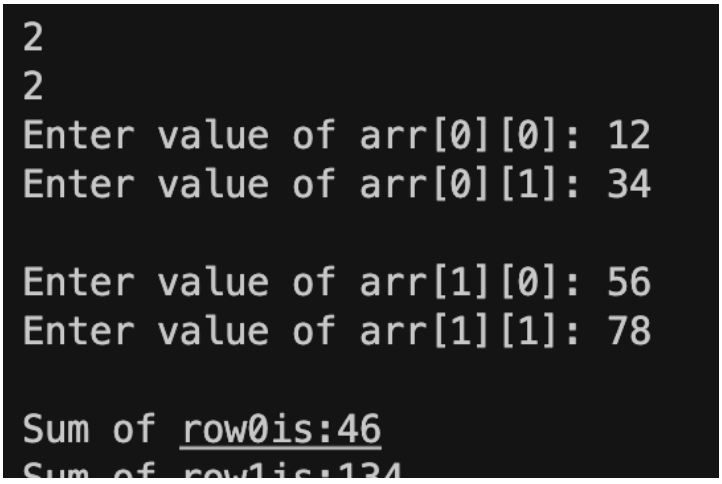
```
        sum=0;
    }
}
```

Output:-

```
Enter value of arr[0][0]: 12
Enter value of arr[0][1]: 34
Enter value of arr[0][2]: 56

Enter value of arr[1][0]: 67
Enter value of arr[1][1]: 89
Enter value of arr[1][2]: 1011

Enter value of arr[2][0]: 23
Enter value of arr[2][1]: 34
Enter value of arr[2][2]: 35
```

Caption

Test case:-



Caption

Conclusion:-

Employing nested loops, the program systematically calculates the sum of elements in each row of the matrix, offering a comprehensive view of row-wise summation.

Experiment No:13

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

Theory:-

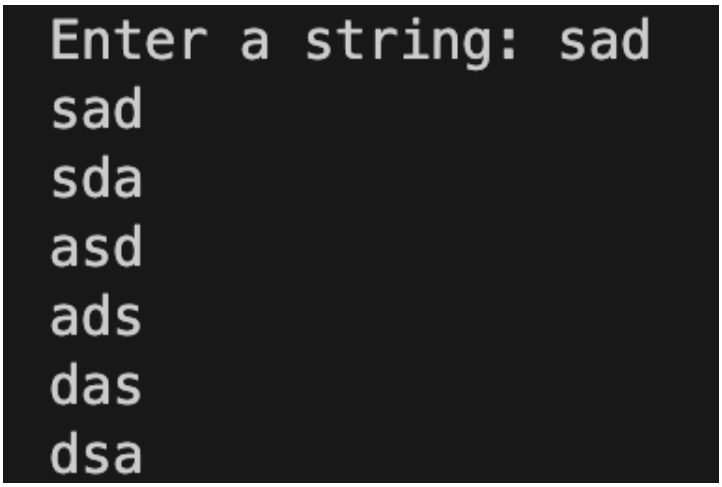
- **The program generates all possible permutations of a string.**
- **It employs recursive algorithms or iterative methods to systematically rearrange characters and produce different permutations.**

Code:-

```
// Write a program to generate all possible permutations of a
string.

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

// function to swap 2 chars
void swap(char &a, char &b)
{
    char temp = a;
    a = b;
    b = temp;
}

// function to generate permunations
void generatePermutations(string str, int left, int right)
{
    if (left == right)
    {
        cout << str << endl;
    }
    else
    {
        for (int i = left; i <= right; ++i)
        {

            swap(str[left], str[i]);
```

```
            generatePermutations(str, left + 1, right);

            swap(str[left], str[i]);
        }
    }
}

int main()
{
    // input a string from user
    string input;

    cout << "Enter a string: ";
    cin >> input;

    int n = input.length();

    // calling function to generate the permutations of given
string
    generatePermutations(input, 0, n - 1);

    return 0;
}
```

Output:-



Caption

Test case:-

```
Enter a string: piyush
piyush
piyuhs
piysuh
piyshu
piyhsu
piyhus
piuysh
piuyhs
piusyh
piushy
piuhsy
piuhys
```

Caption

Conclusion:-

Utilizing recursive or iterative techniques, the program successfully generates and displays all possible permutations of the given string, revealing the various arrangements of its characters.

Experiment No:14

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

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

Theory:-

- **The program uses nested loops to iterate through rows and columns, determining when to print '*' or space.**
- **Conditions check if the current position is on the boundary (first or last row or column) to print '*', otherwise, it prints a space.**

Code:-

```
/* Create a C++ program to print the following pattern:

*****

*  *

*  *

*  *

***** */


#include <iostream>
using namespace std;

int main()
{
```

```
    cout<<"\n";

    for (int i = 1; i <=1; i++)
    {
        for (int j = 1; j <=5; j++)
        {
           cout<<"*";
        }
        cout<<"\n\n";
    }

    for (int k = 1; k <=3; k++)
    {
        for (int l = 1; l <=2 ;l++)
        {
           cout<<"*"<<"  ";
        }
        cout<<"\n\n";

    }

     for (int m = 1; m <=1; m++)
    {
        for (int n = 1; n <=5; n++)
        {
           cout<<"*";
        }

    }
 return 0;
}
```

Output:-

Conclusion:-

- **Through systematic iteration, the program successfully generates a pattern of '*' and spaces, forming a rectangular shape.**
- **The conditional statements ensure the correct placement of '*' on the boundary, producing the intended pattern.**

Experiment No:15

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

1

232

34543

4567654

34543

232

1

Theory:-

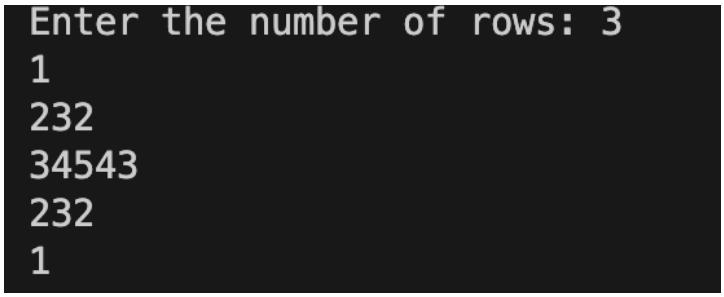
- **The program utilizes nested loops to create a pyramid pattern of numbers.**
- **It systematically generates each row, incrementing and decrementing numbers for the desired output.**

Code:-

```
/* Write a C++ program to display the following pattern:
1
232
34543
4567654
34543
232
1  */

#include <iostream>
using namespace std;

int main() {
    int n;
```

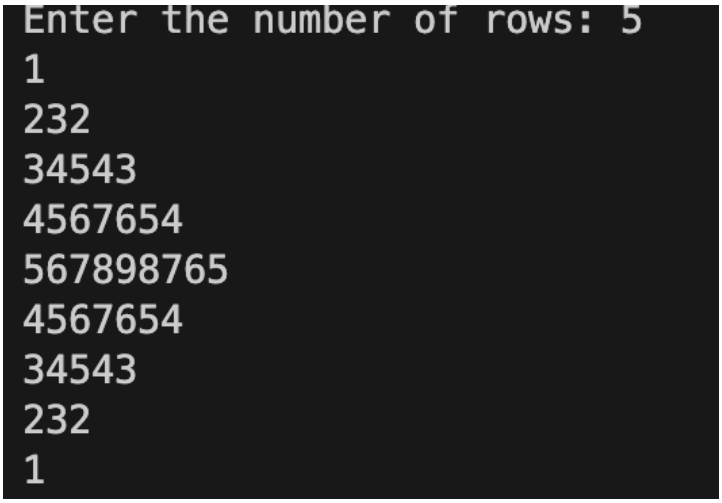
```
    cout << "Enter the number of rows: ";
    cin >> n;

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

        for (int k = i; k <= 2 * i - 1; k++) {
            cout << k;
        }

        for (int l = 2 * i - 2; l >= i; l--) {
            cout << l;
        }

        cout << endl;
    }

    for (int i = n - 1; i >= 1; i--) {
        for (int j = 1; j <= n - i; j++) {
            cout << "";
        }

        for (int k = i; k <= 2 * i - 1; k++) {
            cout << k;
        }

        for (int l = 2 * i - 2; l >= i; l--) {
            cout << l;
        }

        cout << endl;
    }

    return 0;
}
```

Output:-



Caption

Test case:-



Caption

Conclusion:-

- **Through controlled iteration, the program successfully constructs a symmetrical pyramid pattern.**

- **The pattern showcases ascending and descending sequences, forming a visually appealing design.**

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++. Your program 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.

Theory:-

- **he program employs an object-oriented approach in C++ to develop an inventory management system.**
- **The Product class encapsulates essential attributes such as Product ID, Product Name, Price, and Quantity in stock, initializing them through a parameterized constructor.**

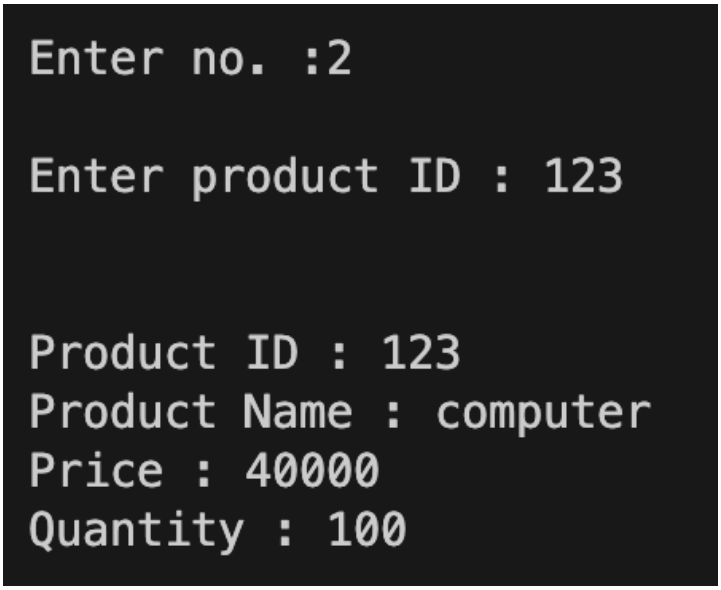
Code:-

```
/*  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.
*/

#include<iostream>
#include<unistd.h>

using namespace std;

class Product
{
private:
   int  product_ID;
   string product_Name;
   float price;
   int Quantity;

public:
  Product(int id , string pn , float pr , int q)
  {
    product_ID = id;
    product_Name = pn;
    price = pr;
    Quantity =q;
  }

  void setData()
  {
    cout<<"\n";
    cout<<"Enter product ID : ";
    cin>>product_ID;
  }

  void getData ()
  {
    cout<<"\n";

    cout<<"\nProduct ID : "<<product_ID<<"\n";
    cout<<"Product Name : "<<product_Name<<"\n";
    cout<<"Price : "<<price<<"\n";
    cout<<"Quantity : "<<Quantity<<"\n";

  }

};


int main()
{
```

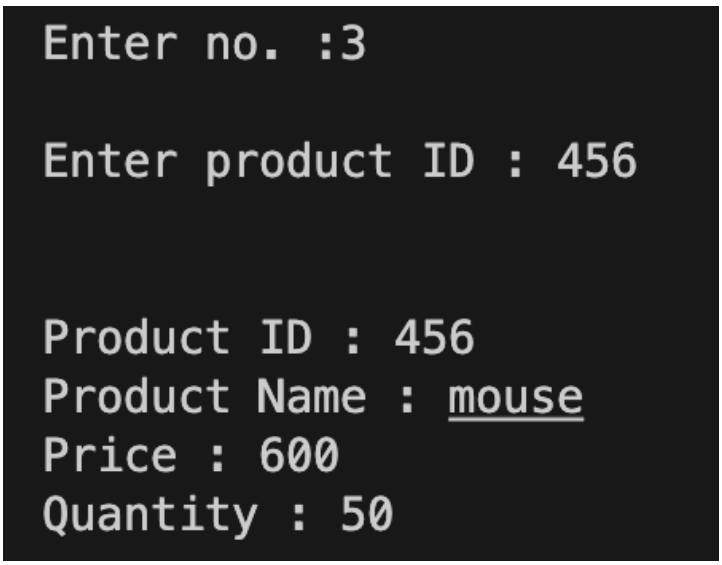
```
  int n;
  Product p1(123,"shampoo",30,50),
          p2(456,"computer",40000,100),
          p3(789,"mouse",600,50),
          p4(901,"moniter",5000,50),
          p5(234,"keyboard",1500,50);

  cout<<"\n\nEnter no. :";
  cin>>n;
  switch (n)
  {
   case 1:
    p1.setData();
    sleep(1);
    p1.getData();
    break;

    case 2:
    p2.setData();
    sleep(1);
    p2.getData();
    break;

    case 3:
    p3.setData();
    sleep(1);
    p3.getData();
    break;

    case 4:
    p4.setData();
    sleep(1);
    p4.getData();
    break;

    case 5:
    p5.setData();
    sleep(1);
    p5.getData();
    break;

    default:
    {
      cout<<"Invalid input";
      break;
    }
  }
return 0;
}
```

Output:-



Caption

Test case:-



Caption

Conclusion:-

- **Utilizing object-oriented principles enhances code modularity and readability in managing the store's inventory.**
- **The parameterized constructor efficiently initializes product attributes, ensuring accurate representation and ease of use in the inventory system.**

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:-

- **The program utilizes a class-based approach in C++ for managing student records.**
- **The Student class encapsulates attributes like name, roll number, and marks, implementing methods to display details, add new students, and calculate the average marks**

Code:-

```
/* Design a C++ 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.

*/

#include<iostream>
using namespace std;

class Student
{
  private:
  int p;
  int c;
  int m;
```

```
  string Student_name;
  int rollno;
  float marks;

  public:

    void setData()
     {
       cout<<"\n\n";

       cout<<"\nName of the student : ";
       cin>>Student_name;

       cout<<"Roll no. of the student : ";
       cin>>rollno;

       cout<<"Enter Marks Of Physics : ";
       cin>>p;

       cout<<"Enter Marks Of Chemistry : ";
       cin>>c;

       cout<<"Enter Marks Of Math : ";
       cin>>m;


     }

     void gitData()
     {
       cout<<"\n";

       marks=(p+c+m)/3;
       cout<<"average marks is : "<<marks;

     }


};

int main()
{
  Student s1,s2,s3;

  s1.setData();
  s1.gitData();

  s2.setData();
```

```
  s2.gitData();

  s3.setData();
  s3.gitData();
}
```

Output:-

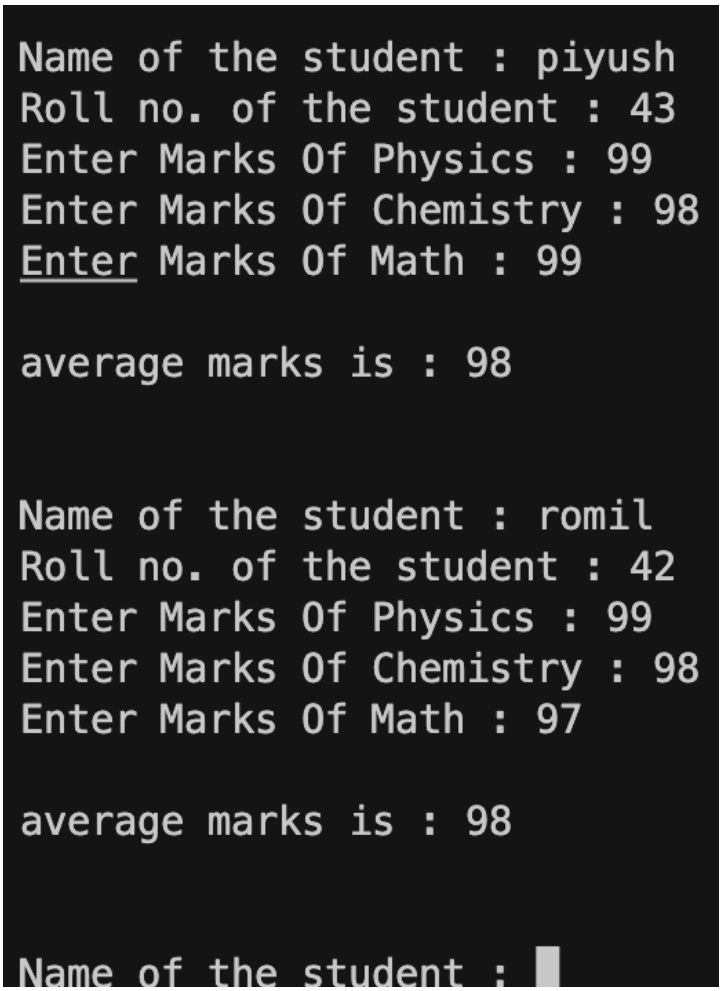
Test case:-

```
Name of the student : piyush
Roll no. of the student : 78
Enter Marks Of Physics : 45
Enter Marks Of Chemistry : 4
Enter Marks Of Math : 55

average marks is : 34


Name of the student : romil
Roll no. of the student : 56
Enter Marks Of Physics : 3
Enter Marks Of Chemistry : 34
Enter Marks Of Math : 45

average marks is : 27


Name of the student :
```

Caption

Conclusion:-

- **The class-based structure enhances code organization and maintenance for efficient student**



Caption

record management.

- **The implemented methods provide essential functionalities, ensuring ease of use and accuracy in maintaining and processing student information.**

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:-

- **The program employs a class-based design in C++ to create a basic calculator.**
- **The Calculator class encompasses methods for addition, subtraction, multiplication, and division, offering a modular structure for arithmetic operations.**

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 Calculater
{
private:
   float n1, n2;

   int choice;

public:
   void setData()
```

```
    {
        cout << "\n";

        cout << "\nEnter first number : ";
        cin >> n1;

        cout << "Enter second number : ";
        cin >> n2;

        cout << "Enter your choice :(1.'+' 2.'-' 3.'*' 4.'/' 5.'%') :
";
        cin >> choice;
    }

    void getData()
    {
        switch (choice)
        {
        case 1:
            cout << "\nSum of " << n1 << " + " << n2 << " is : " << n1
+ n2;
            break;

        case 2:
            cout << "\nsubtraction of " << n1 << " - " << n2 << " is :
" << n1 - n2;
            break;

        case 3:
            cout << "\nMultipication of " << n1 << " * " << n2 << " is
: " << n1 * n2;
            break;

        case 4:
            cout << "\nDivision of " << n1 << " / " << n2 << " is : "
<< n1 / n2;
            break;

        case 5:
            cout << "\nModulus of " << n1 << " % " << n2 << " is : "
<< int(n1) % int(n2);
            break;

        default:
            cout << "Invalid Choice!";
            break;
        }
    }
};
```

```cpp
int main()
{
   do
   {
      int cho;
      bool again = true;

      Calculater c1;

      c1.setData();
      c1.getData();

      do
      {
         cout << "\n\nYou want calculate another no. : ('1' for Yes
or '2' for NO ) : ";
         cin >> cho;

         if (cho != 1 && cho != 2)
         {
            cout << "invalid input!";
         }

      } while (cho != 1 && cho != 2);

      if (cho == 2)
      {
         cout << "Thank u :)";
         break;
      }
   } while (true);

   return 0;
}
```

Output:-

```
Enter first number : 2
Enter second number : 3
Enter your choice :(1.'+' 2.'-' 3.'*' 4.'/' 5.'%') : 3

Multipication of 2 * 3 is : 6

You want calculate another no. : ('1' for Yes or '2' for NO ) : 1


Enter first number :
```

Caption

Test case:-

```
Enter first number : 40
Enter second number : 50
Enter your choice :(1.'+' 2.'-' 3.'*' 4.'/' 5.'%') : 5

Modulus of 40 % 50 is : 40

You want calculate another no. : ('1' for Yes or '2' for NO ) : 2
Thank u :)%
```

Caption

Conclusion:-

- **The class-based approach provides a clear and organized structure for a functional calculator program.**
- **User inputs and selected operations are seamlessly processed, making the calculator user-friendly and versatile.**

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:-

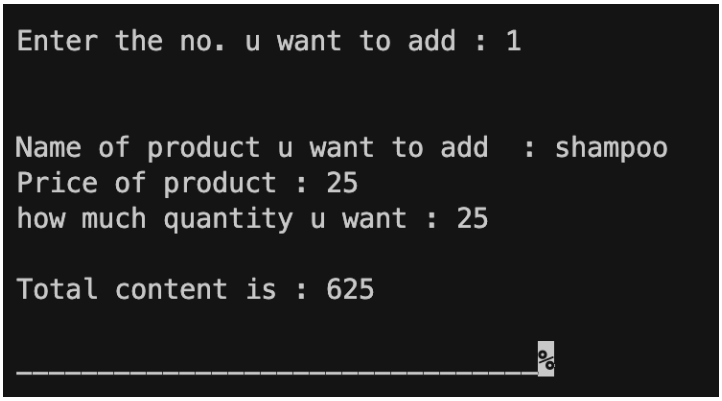
- **The program utilizes C++ and object-oriented principles to simulate an online shop.**
- **The Product class encapsulates attributes (name, price, quantity) and methods (add to cart, calculate total cost, display cart) for effective online shopping simulation.**

Code:-

```
/*
   Design 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.
*/

#include <iostream>
using namespace std;

class Product
{
 private:
   string product_name;
   int price;
   int quantity;
   int cal;
```

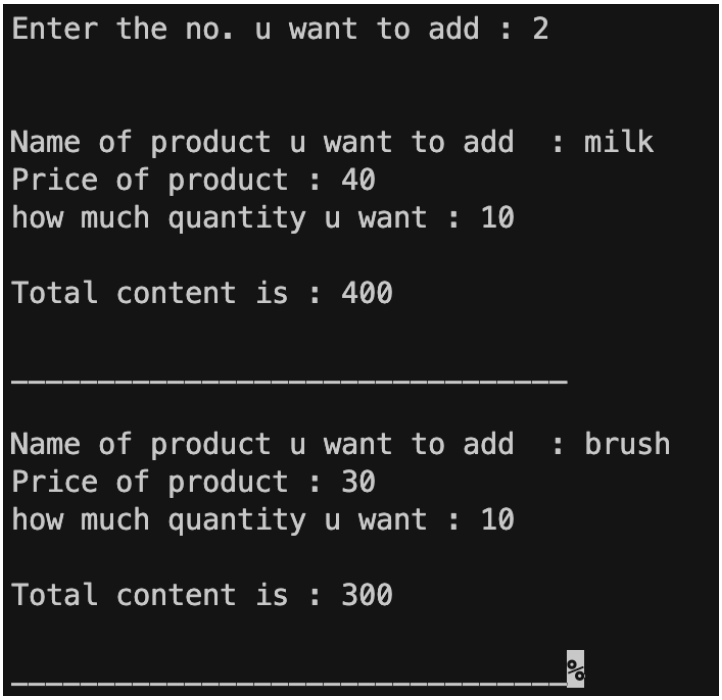
```
    public:

    void addProduct()
    {
        cout<<"\n\nName of product u want to add  : ";
        cin>>product_name;

        cout<<"Price of product : ";
        cin>>price;

        cout<<"how much quantity u want : ";
        cin>>quantity;

    }

    void calculate()
    {
        cal=price*quantity;
    }

    void displayContent()
    {

        cout<<"\nTotal content is : "<<cal;
        cout<<"\n\n_____________________________________";
    }
};

int main()
{

    int n;

    cout<<"\n\nEnter the no. u want to add : ";
    cin>>n;

    for (int i = 1; i <=n; i++)
    {
        Product  p1;

        p1.addProduct();
        p1.calculate();
        p1.displayContent();
    }


}
```

Output:-



Caption

Test case:-



Caption

Conclusion:-

- **Object-oriented design enhances the program's structure, allowing for easy expansion and maintenance.**
- **The implemented methods facilitate a seamless online shopping experience, incorporating essential functionalities for managing product interactions and displaying cart contents.**

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:-

- **The program, written in C++, employs a Student class for managing student grades.**
- **Attributes include student name and an array for grades. Methods for adding grades, calculating the average, and displaying information are implemented. Constructors and destructors manage resource initialization and release.**

Code:-

```
#include <iostream>
#include <unistd.h>

using namespace std;

class Student
{
    private:
        int roll_no ;
        string name , grade;
        float marks , average;

    public:
        Student(){
            average = 0;
            marks = 0;
            name = "";
            grade = "";
```

```
            roll_no = 0;
        };
        void getinfo()
        {
            cout<<"Enter Student name : ";
            cin>>name;
            cout<<"\nEnter Roll No. : ";
            cin>>roll_no;

            for(int i = 1 ; i <= 5 ; i++)
            {
                rerun:
                cout<<"Enter subject "<<i<<" marks : ";
                cin>>marks;

                if(marks > 100 || marks < 0)
                {
                    cout<<"marks should not exceed 100 and Should
not be negative :) \n";
                    goto rerun;
                }

                average += marks;
            }
        }

        void displayinfo()
        {
            cout<<"NAME : "<<name<<"\n";
            cout<<"ROLL NO. : "<<roll_no<<"\n";
            cout<<"Total marks(out of 500) : "<<average<<"\n";

            average /= 5.00;
            if(average >= 85 && average < 95)
            {
                cout<<"PERCENTAGE : "<<average<<"% with GRADE : A";
            }
            else if(average >= 95)
            {
                cout<<"PERCENTAGE : "<<average<<"% with GRADE :
A+";
            }
            else if(average >= 75 && average < 85)
            {
                cout<<"PERCENTAGE : "<<average<<"% with GRADE : B";
            }
            else if(average < 75 && average >= 60)
            {
                cout<<"PERCENTAGE : "<<average<<"% with GRADE : C";
```

```
            }
            else if(average < 60 && average > 33)
            {
                cout<<"PERCENTAGE : "<<average<<"% with GRADE : D";
            }
            else
            {
                cout<<"PERCENTAGE : "<<average<<"% and failed class
with GRADE : F";
            }
            sleep(2);
        }

        ~Student(){

        }

};


int main()
{
    int n;
    char y;

    cout<<"\nEnter the number of students you want to enter details
of? \n";
    cin>>n;

    Student stud[n];

    for(int i = 0 ; i<n ; i++)
    {
        stud[i].getinfo();
    }


    cout<<"Do you want to display data?(y/n)";
    cin>>y;

    if(toupper(y) == 'Y')
    {
        for(int i = 0 ; i<n ; i++)
        {
            cout<<"\n__________\n";
            cout<<"STUDENT "<<i+1<<"\n";
            stud[i].displayinfo();

        }
```

```
    }
    else
    {
        return 0;
    }
}
```

Output:-

```
2
Enter Student name : prem

Enter Roll No. : 2
Enter subject 1 marks : 56
Enter subject 2 marks : 45
Enter subject 3 marks : 3
Enter subject 4 marks : 4
Enter subject 5 marks : 56
Enter Student name : patil

Enter Roll No. : 44
Enter subject 1 marks : 67
Enter subject 2 marks : 78
Enter subject 3 marks : 78
Enter subject 4 marks : 99
Enter subject 5 marks : 78
Do you want to display data?(y/n)y


__________
STUDENT 1
NAME : prem
ROLL NO. : 2
Total marks(out of 500) : 164
PERCENTAGE : 32.8% and failed class with GRADE : F
__________
STUDENT 2
NAME : patil
ROLL NO. : 44
Total marks(out of 500) : 400
PERCENTAGE : 80% with GRADE : B%
```

Caption

Test case:-

```
Enter the number of students you want to enter details of?
2
Enter Student name : piyush

Enter Roll No. : 43
Enter subject 1 marks : 99
Enter subject 2 marks : 98
Enter subject 3 marks : 97
Enter subject 4 marks : 99
Enter subject 5 marks : 99
Enter Student name : romil

Enter Roll No. : 42
Enter subject 1 marks : 99
Enter subject 2 marks : 99
Enter subject 3 marks : 98
Enter subject 4 marks : 07
Enter subject 5 marks : 96
Do you want to display data?(y/n)y

__________
STUDENT 1
NAME : piyush
ROLL NO. : 43
Total marks(out of 500) : 492
PERCENTAGE : 98.4% with GRADE : A+
__________
STUDENT 2
NAME : romil
ROLL NO. : 42
Total marks(out of 500) : 399
PERCENTAGE : 79.8% with GRADE : B
```

Caption

Conclusion:-

- **The class-based structure facilitates efficient management of student grades.**

- **Constructors and destructors ensure proper resource handling, enhancing the program's reliability and resource efficiency.**

THANK YOU