LAB EXERCISES :1

1. First C++ Program: Hello World o Write a simple C++ program to display "Hello, World!". o Objective: Understand the basic structure of a C++ program, including #include, main(), and cout.

```
#include<iostream>

Using namespace std;

Iint main()

{

Cout <<”Hello Wolrd”;

Return 0;

}
```

2. Basic Input/Output o Write a C++ program that accepts user input for their name and age and then displays a personalized greeting. o Objective: Practice input/output operations using Cin and cout.

```
#include<iostream>

Using namespace std;

Int main()

{

String name;

Int age;

cout <<”Enter the Name :” <<endl;

Cin >>name;

cout <<”Enter the Age:”<<endl;

Cin >>age;

Cout   <<”Name:” <<name <<endl;

cout <<”Age:” <<age <<endl;

Return 0;

}
```

```
OutPut-------------

Enter the  name: Ravi

Enter Age: 26

Name:Ravi

Age:26
```

3. POP vs. OOP Comparison Program

- Write two small programs: one using Procedural Programming (POP) to calculate the area of a rectangle, and another using Object-Oriented Programming (OOP) with a class and object for the same task.

<u>POP</u>

```
#include<stdio.h>

int main()

 {

float r, area;

printf("\nEnter the radius of the circle: ");
scanf("%f", &r);

area = 3.14 * r * r;
printf("\nArea of Circle: %.2f\n", area);

return 0;
}
```

```
OutPut------------------

Enter the Radius Of the Circle : 5

Area of Circle: 75.25
```

```
/*Formula Area=π×r×r*/

#include<stdio.h>

float Area(float r)

{

   float area = 3.14 * r * r;

   return area;

}

int main()

{

    float num,result;

    printf("\n Enter The Value:");

    scanf("%f",&num);

    result = Area(num);

    printf("\n Area of Circle :%.2f",result);

    return 0;

}

Output----------------------

Enter the Value: 5

75.25

-----------------------------------------------------------------------------
```

❖ Objective: Highlight the difference between POP and OOP approaches.

Procedural Oriented Programming	Object Oriented Programming
1.POP organizes code around functions and data	**2.OOP** organizes code around objects and classes
2.Data is shared globally and can be accessed from any function.	2.Data is encapsulated (hidden inside a class), and is therefore private.
3.Pop main code is about reusability.	3. More secure and reusable

4. Setting Up Development Environment o Write a program that asks for two numbers and displays their sum. Ensure this is done after setting up the IDE (like Dev C++ or Code Blocks). o Objective: Help students understand how to install, configure, and run programs in an IDE.

```
#include<iostream>

Using namespace std;

Int main()

{

Int num1,num2,ans;

Cout <<"Enter Num1:";

Cin>>num1;

Cout <<"Enter Num2:";

Cin>>num2;

ans = num1 +  num2;

cout <<"Sum is : "<<ans <<endl;

return  0;

}
```

Output-------------------------------

```
Enter Num1: 10

Enter Num2: 10

Sum is : 20
```

LAB EXERCISES: 2.

Variables and Constants

❖ Write a C++ program that demonstrates the use of variables and constants. Create variables of different data types and perform operations on them.

```
#include<iostream>
using namespace  std;
int main()
{

   int roll;
   string name;
   long int fees;
   float percentage;

   cout<<"Enter The Roll Number:";
   cin>>roll;
   cout<<"Enter The Name:";
   cin>>name;
   cout<<"Enter The Fees:";
   cin>>fees;
   cout<<"Enter The Percentage:";
   cin>>percentage;

   cout<<"\n";
   cout<<"Roll Number Is :"<<roll<<endl;
   cout<<"Name is :"<<name<<endl;
   cout<<"Fees is :"<<fees<<endl;
   cout<<"Parcentage is :"<<percentage<<endl;
   return 0;
}
```

Output---

```
Enter The Roll Number:101
Enter The Name:ravi
Enter The Fees:58000
Enter The Percentage:78.98

Roll Number Is :101
Name is :ravi
Fees is :58000
Parcentage is :78.98
```

- Objective: Understand the difference between variables and constants. 2. Type Conversion
    - Variable: A value that **can change** during program execution.

    - Constants : A value that **cannot be changed** once defined.

    - **Implicit Conversion**

    Example int a = 5;
    float b = a; // int to float automatically

    - **Explicit Conversion**

    Example : float pi = 3.14;
    int x = (int)pi; // Only the integer part remains => x = 3

- Write a C++ program that performs both implicit and explicit type conversions and prints the results.

```
#include<iostream>
using namespace std;
   int main() {
   int Value1 = 10;
   float Value2;
   Value1 = Value2;
   cout << "Implicit Type Casting (int to float):" << endl;
   cout << "int value1: " << intValue1 << endl;
   cout << "float value (after implicit conversion): " << floatValue << endl;


   float original = 9.78;
   int converteds;

   convertedInt = (int)original;

   cout << "\nExplicit Type Casting (float to int):" << endl;
   cout << "Original float: " << original << endl;
   cout << "Converted  (after explicit cast): " << converteds << endl;

   return 0;
}
```

Output---

```
Implicit Type Casting (int to float):
int value: 10
float value (after implicit conversion): 10

Explicit Type Casting (float to int):
Original float: 9.78
Converted int (after explicit cast): 9
```

❖ Write a C++ program that demonstrates arithmetic, relational, logical, and bitwise operators.

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

  int main() {
  int a = 10, b = 5;

  // ----- Arithmetic Operators -----
  cout << "----- Arithmetic Operators -----" << endl;
  cout << "a + b = " << (a + b) << endl;
  cout << "a - b = " << (a - b) << endl;
  cout << "a * b = " << (a * b) << endl;
  cout << "a / b = " << (a / b) << endl;
  cout << "a % b = " << (a % b) << endl;

  // ----- Relational Operators -----
  cout << "\n----- Relational Operators -----" << endl;
  cout << "a == b: " << (a == b) << endl;
  cout << "a != b: " << (a != b) << endl;
  cout << "a > b : " << (a > b) << endl;
  cout << "a < b : " << (a < b) << endl;
  cout << "a >= b: " << (a >= b) << endl;
  cout << "a <= b: " << (a <= b) << endl;

  // ----- Logical Operators -----
  cout << "\n----- Logical Operators -----" << endl;
  bool x = true, y = false;
  cout << "x && y: " << (x && y) << endl;
  cout << "x || y: " << (x || y) << endl;
  cout << "!x    : " << (!x) << endl;


  return 0;  }
```

❖ Perform operations using each type of operator and display the results.

```
#include<iostream>
Using namespace std;
Int main()
{
    Int Num1,Num2,Ans;
    Cout<<"Enter  the Num1 :";
    Cin>>Num1;
```

❖ Objective: Reinforce understanding of different types of operators in C++.

Operator Type	Description	Example
1. Arithmetic	Performs basic math operations	`+, -, *, /, %`
2. Relational	Compares two values	`==, !=, <, >, <=, >=`
3. Logical	Combines conditional expressions	`&&, `` ` ``
4. Assignment	Assigns values to variables	`=, +=, -=, *=, /=, %=`
5. Increment/Decrement	Increases or decreases value by 1	`++, --`
6. Bitwise	Operates on bits	`&`
7. Conditional (Ternary)	Short-hand for if-else	`condition ? expr1 : expr2`

LAB EXERCISES:3

1. Grade Calculator o Write a C++ program that takes a student's marks as input and calculates the grade based on if-else conditions.

```
#include<iostream>
using namespace std;
int main()
{
   int num1,num2,ans;
   char operators;
   cout <<"\n Enter the Number 1  :";
   cin >>num1;
```

```
    cout <<"\n Enter the Number 2  :";
    cin >>num2;
    cout <<"\n Enter your Choice [+][-][*][/]";
    cin >>operators;

    if (operators == '+')
    {
      cout <<"Addtion is :" <<num1 + num2 <<endl;
    }
    else if (operators == '-')
    {
       cout <<"Subtraction is :" <<num1 - num2 <<endl;
    }
    else if (operators == '*')
    {
       cout <<"Multiplication is :" <<num1 * num2 <<endl;
    }
    else if (operators == '/')
    {
       cout <<"Division is :" <<(float)num1 / (float)num2 <<endl;
    }
Return  0;
}
```

2. Write a C++ program to display the multiplication table of a given number using a for loop.

```
#include<iostream>
using namespace std;
int main()
{
    int num,i;
    cout <<"\n Enter the Number ";
    cin >>num;
    for ( i = 1; i <= num; i++)
    {
       cout <<num <<" X " <<num <<" = " <<i*num <<endl;
    }
    return 0;
}
```

3. Write a program that prints a right-angled triangle using stars (*) with a nested loop.

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

int main() {
  int rows;

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


  for (int i = 1; i <= rows; i++) {

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

  return 0;
}
```

LAB EXERCISES: 4

1. Simple Calculator Using Functions o Write a C++ program that defines functions for basic arithmetic operations (add, subtract, multiply, divide). The main function should call these based on user input

```
#include<iostream>
using namespace std;
int add(int a,int b)
{
  int ans = a+b;
  return ans;
}
int subtra(int a,int b)
{
  int ans = a-b;
  return ans;
}
int multi(int a,int b)
{
    int ans = a*b;
    return ans;
}
int divi(float a,float b)
{
    int ans = a/b;
    return ans;
}
int main()
{
    int num1,num2,sum,sub,mul,div;
    cout <<"Enter the num1 :";
    cin >>num1;
    cout <<"Enter the num2 :";
    cin >>num2;
    sum = add(num1,num2);
    sub = subtra(num1,num2);
    mul = multi(num1,num2);
    div = divi((float)num1,(float)num2);
    cout <<"The Addition is:"<<sum <<endl;
    cout <<"The Substraction is:" <<sub <<endl;
    cout <<"The Multiplication  is:" <<mul <<endl;
    cout <<"The Division is:" <<div <<endl;

    return 0;
}
```

2. Factorial Calculation Using Recursion o Write a C++ program that calculates the factorial of a number using recursion.

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

int fact(int num)
{
    if(num!=0)
    {
        return num * fact(num-1);
    }
    else
    {
        return 1;
    }
}
```

```
int main()
{
   int num;
   cout<<"Enter the number = ";
   cin>>num;//5
   int answer = fact(num);
   cout<<"\nFactorial of "<<num<<" is = "<<answer;
   return 0;
}
```

LAB EXERCISES: 5

1. Array Sum and Average Write a C++ program that accepts an array of integers, calculates the sum and average, and displays the results. Objective: Understand basic array manipulation.

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

int main() {
   int a[100], n, sum = 0;
   float average;

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

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


   average = (float)sum / n;


   cout << "Sum = " << sum << endl;
   cout << "Average = " << average << endl;

   return 0;
}
```

2. Matrix Addition . Write a C++ program to perform matrix addition on two 2x2 matrices.
   Objective: Practice multi-dimensional arrays.

```
include <iostream>
using namespace std;

int main() {
  int A[2][2], B[2][2], sum[2][2];

  cout << "Enter elements of Matrix A (2x2):" << endl;
  for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 2; j++) {
      cout << "A[" << i << "][" << j << "]: ";
      cin >> A[i][j];
    }
  }

  cout << "\nEnter elements of Matrix B (2x2):" << endl;
  for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 2; j++) {
      cout << "B[" << i << "][" << j << "]: ";
      cin >> B[i][j];
    }
  }

  for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 2; j++) {
      sum[i][j] = A[i][j] + B[i][j];
    }
  }

  cout << "\nMatrix after Addition (A + B):" << endl;
  for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 2; j++) {
      cout << sum[i][j] << " ";
    }
    cout << endl;
  }

  return 0;
}
```

3. String Palindrome Check . Write a C++ program to check if a given string is a palindrome (reads the same forwards and backwards). Objective: Practice string operations.

```
#include<iostream>

using namespace std;

int main()

{

   int num;

   cout <<"\n Enter the num:";

   cin>>num;

   int temp = num;

   int digit,rem,rev;

   while (num!=0)

   {

      rev = num % 10;

      rev = rev * 10 + rem;

      num = num / 10;

   }

   if (rev == temp)

   {

      cout <<"Is palindrome" <<temp <<endl;

   }

   else

   {

      cout <<"Is Not palindrome" <<temp <<endl;

   }
   return 0;

}
```

LAB EXERCISES: 6

1. Class for a Simple Calculator o Write a C++ program that defines a class Calculator with functions for addition, subtraction, multiplication, and division. Create objects to use these functions. o Objective: Introduce basic class structure.

```
#include<iostream>
using namespace std;
class claculator
{
    public:
    int add(int a,int b)
    {
      cout <<"The Addition is:" <<a+b <<endl;
    }
    int sub(int a,int b)
    {
       cout <<"The Subtraction is:" <<a-b <<endl;
    }
    int mul(int a,int b)
    {
        cout <<"The Multiplication is:" <<a*b <<endl;
    }
    float div(float a,float b)
    {
        cout <<"The Division is:" <<a/b <<endl;
    }
};
int main()
{
    claculator c1;
    c1.add(10,10);
    c1.sub(40,10);
    c1.mul(10,20);
    c1.div(10,5);
return 0;
}
```

2. . Class for Bank Account . Create a class Bank Account with data members like balance and member functions like deposit and withdraw. Implement encapsulation by keeping the data members private. Objective: Understand encapsulation in classes.

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

class BankAccount {
private:
  int balance;

public:
  BankAccount() {
    balance = 0;
  }


  void deposit(int amount) {
    balance = balance + amount;
    cout << "Deposited: " << amount << endl;
  }


  void withdraw(int amount) {
    if (amount <= balance) {
      balance = balance - amount;
      cout << "Withdrawn: " << amount << endl;
    } else {
      cout << "Not enough balance!" << endl;
    }
  }


  void showBalance() {
    cout << "Current Balance: " << balance << endl;
  }
};

int main() {
  BankAccount myAcc;
```

```
    myAcc.deposit(1000);
    myAcc.withdraw(500);
    myAcc.withdraw(700);
    myAcc.showBalance();

    return 0;
}
```

3. Inheritance Example .Write a program that implements inheritance using a base class Person and derived classes Student and Teacher. Demonstrate reusability through inheritance.
   Objective: Learn the concept of inheritance.

```
#include<iostream>
using namespace std;
class Person
{
    public:
    string name="Ravi Arya";
};
class student :public Person
{
    public:
    int age=26;
    int RollNo=101;
    string Behavior="Good";
    void display()
    {
        cout <<"\n Student Name :" <<name;
        cout <<"\n Student Age  :" <<age;
        cout <<"\n Student Roll No:" <<RollNo;
        cout <<"\n Student Behavior" <<Behavior;
    }
};
int main()
{
    student s1;
    s1.display();
    return 0;
}
```

Output---

```
Student Name :Ravi Arya
 Student Age  :26
 Student Roll No:101
 Student Behavior : Good
```