LAB 1

OBJECTIVE: Writing Simple Programs in C++.

THEORY:

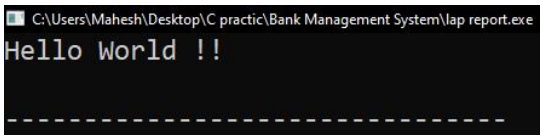
C++ (or "C-plus-plus") is a generic programming language for building software. It's an object-oriented language. In other words, it emphasizes using data fields with unique attributes (a.k.a. objects) rather than logic or functions. A common example of an object is a user account on a website.

PROGRAMS:

a. WAP to display the message "Hello World" using C++.

```
#include<iostream>
using namespace std;
int main(){
        cout<<"Hello World !!"<<endl;
        return 0;
}
```

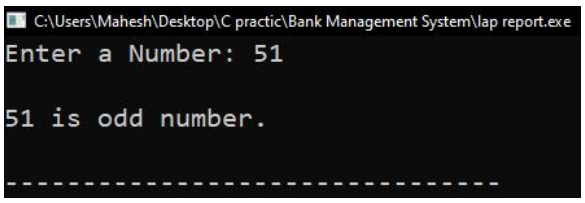
Output:



b. WAP to input a number and display Even or Odd using C++.

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

int main(){
        int num;

        cout<<"Enter a Number: "; cin>>num;
        cout<<endl;
        if(num%2==0) cout<<num<<" is even number."<<endl;
        else cout<<num<<" is odd number."<<endl;

        return 0;
}
```

Output:

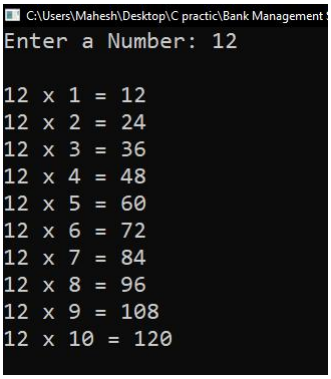


c. WAP to input a number and display its table upto 10 terms using C++.

```
#include<iostream>
using namespace std;
int main(){
        int num;
        cout<<"Enter a Number: "; cin>>num;
        cout<<endl;
```

```
    for(int i=1; i<=10; i++)
        cout<<num<<" x "<<i<<" = "<<(num*i)<<endl;
    return 0;
}
```

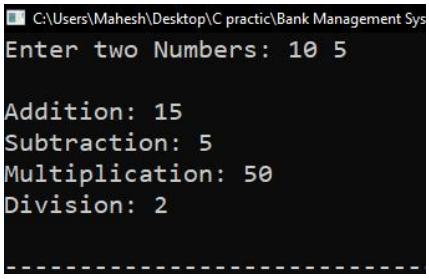
Output:



d. WAP to input two numbers and display their sum, difference, product and division.

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

int main(){
    int n1,n2;

    cout<<"Enter two Numbers: "; cin>>n1>>n2;

    cout<<"\nAddition: "<<(n1+n2)<<endl;
    cout<<"Subtraction: "<<(n1-n2)<<endl;
    cout<<"Multiplication: "<<(n1*n2)<<endl;
    cout<<"Division: "<<(n1/n2)<<endl;

    return 0;
}
```

Output:



RESULTS AND DISCUSSION:

The experiment was successful to write simple programs in C programming. This program helps to understand basics of C++ programming language.

CONCLUSION:

This laboratory exercise provided a hands-on experience in C++ program. Students gained practical knowledge of implementing basic in C programming and are now better equipped to undertake more complex programming tasks in the future.

LAB 2

OBJECTIVE: Writing C++ programs using Classes and objects.

THEORY:

A class is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A C++ class is like a blueprint for an object.

- A Class is a user-defined data type that has data members and member functions.
- Data members are the data variables and member functions are the functions used to manipulate these variables together, these data members and member functions define the properties and behaviour of the objects in a Class.
- In the above example of class Car, the data member will be speed limit, mileage, etc, and member functions can be applying brakes, increasing speed, etc.

PROGRAM:

a. Define a class Person with private members for the person's name and age. Write methods to set and get these values.

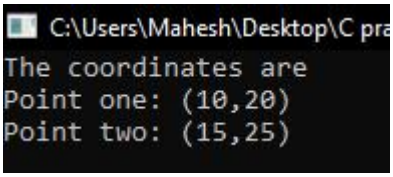
```
#include <iostream>
#include <string>
using namespace std;

class Person {
private:
   string name;
   int age;
public:
   void setDetails(string n,int a) {
      name = n;
      age = a;
   }
   void getDetails() {
      cout<<"Name is: "<<name<<endl;
      cout<<"Age is: "<<age<<endl;
   }
};

int main() {
   Person person;
   person.setDetails("Ramesh",30);
   person.getDetails();
   return 0;
}
```

Output:

b. Create a class Point that represents a point in a 2D space with x and y coordinates. Write methods to set and get the coordinates.

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

class Point {
        double x,y;
   public:
        void setPoint(double X,double Y){
                x=X; y=Y;
        }
        void getPoint(){
                cout<<"("<<x<<","<<y<<")"<<endl;
        }
};
int main() {
        Point p1,p2;
        p1.setPoint(10,20);
        p2.setPoint(15,25);

        cout<<"The coordinates are"<<endl;
        cout<<"Point one: "; p1.getPoint();
        cout<<"Point two: "; p2.getPoint();

        return 0;
}
```

Output:

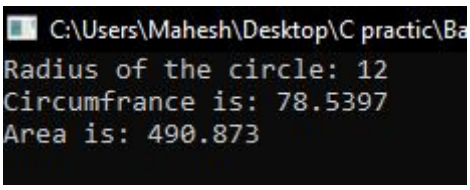


c. Define a class Circle with a member for the radius. Write methods to calculate the circumference and area of the circle.

```
#include <iostream>
#include <string>
#define pi 3.14159
using namespace std;

class Circle {
        double rad;

public:
   Circle(double r):rad(r){}

        void Circumfrance(){
                cout<<"Circumfrance is: "<<(2*pi*rad)<<endl;
        }
        void Area(){
                cout<<"Area is: "<<(pi*rad*rad)<<endl;
```

```
        }
        int getRadius(){ return rad; }
};

int main() {
    Circle c1(12.5);

    cout<<"Radius of the circle: "<<c1.getRadius()<<endl;
    c1.Circumfrance();
    c1.Area();

    return 0;
}
```

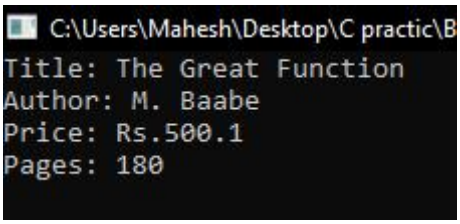
Output:



d. Implement a class Book with members for the title, author, price and number of pages. Include methods to set and get these values.

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

class Book {
    string title;
    string author;
    double price;
    int pages;

public:
    Book(const string& t, const string& a, double p, int pg) : title(t), author(a), price(p), pages(pg) {}

    void setDetails(const string& t, const string& a, double p, int pg) {
        title = t;
        author = a;
        price = p;
        pages = pg;
    }
    void displayBookDetails() const {
        cout << "Title: " << title << "\nAuthor: " << author << "\nPrice: Rs." << price << "\nPages: " << pages << endl;
    }
};

int main() {
    Book myBook("The Great Function", "M. Baabe", 500.10, 180);
    myBook.displayBookDetails();

    return 0;
}
```

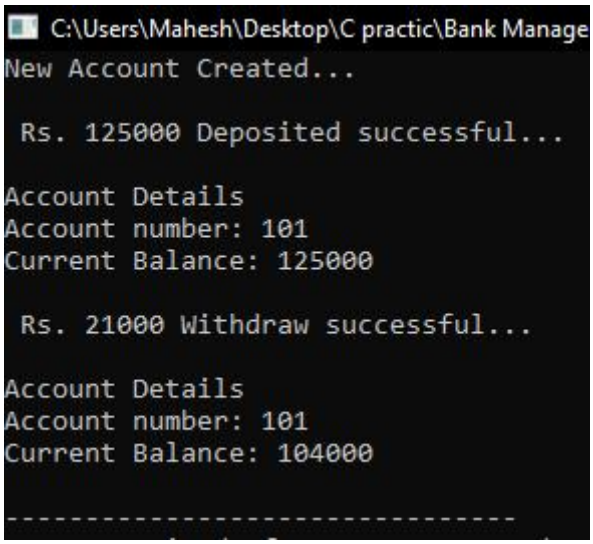
Output:



e. Create a class BankAccount with members for the account number and balance. Write methods to deposit and withdraw money, and to check the balance.

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

class BankAccount {
   int accNumber;
   double balance;

public:
   BankAccount(int accNo):accNumber(accNo), balance(0.00){
                cout<<"New Account Created..."<<endl;
        }

   void depositeAmount(double amt){
                balance += amt;
                cout<<"\n Rs. "<<amt<<" Deposited successful...\n"<<endl;
        }
        void withdrawAmount(double amt){
                if(balance < amt)
                        cout<<"Don't have that much Amount in this account..!!"<<endl;
                else{
                        balance -= amt;
                        cout<<"\n Rs. "<<amt<<" Withdraw successful...\n"<<endl;
                }
        }
        int display(){
                cout<<"Account Details"<<endl;
                cout<<"Account number: "<<accNumber<<endl;
                cout<<"Current Balance: "<<balance<<endl;
        }
};

int main() {
   BankAccount acc1(101);
   acc1.depositeAmount(125000);
   acc1.display();

   acc1.withdrawAmount(21000);
   acc1.display();

   return 0;
}
```

Output:



f. Write a class Date that represents a date with day, month, and year members. Include methods to set and display the date.

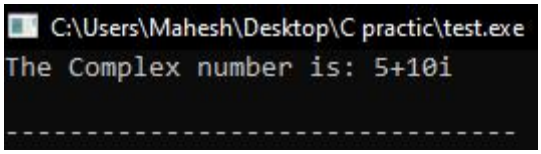
```
#include <iostream>
using namespace std;

class Date {
   string day,month, year;

public:
        void setDate(string y,string m,string d){
                year = y;
                month = m;
                day = d;
        }
        string getDate(){
                string date = year+"/"+month+"/"+day;
                return date;
        }
};

int main() {
   Date date;
   date.setDate("2050","11","26");
   cout<<"\nDate is: "<<date.getDate()<<endl;

   return 0;
}
```

Output:

g. Create a class Complex Number and write methods to set and get the number.

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

class Complex_Number{
        double a,b;
        public:
                void set(double a1, double b1){
                        a = a1; b = b1;
                }
                void get(){
                        cout<<"The Complex number is: "<<a<<"+"<<b<<"i"<<endl;
                }
};

int main(){
        Complex_Number cn;
        cn.set(5,10);
        cn.get();

        return 0;
}
```

Output:



RESULTS AND DISCUSSION:

In this lab, different types of class and object based programming was done in C++ programming language. Students get the practical knowledge about the class and object concept in C++ programming.

CONCLUSION:

This program provides a basic implementation of the class and object concept in C++. It serves as a useful example for understanding how to implement classes as blueprint of the program.

LAB 3

OBJECTIVE:

To illustrate the concept of different types of constructors (default, parameterized and copy) in C++.

THEORY:

In C++, a constructor is a special member function of a class that is automatically called when an object of that class is created. It is used to initialize the object's data members and to allocate resources if necessary. Types of constructor are mentioned below.

1. Default Constructor
   A default constructor is one that takes no arguments. If no constructor is defined in a class, the compiler provides a default constructor.

2. Parameterized Constructor
   A parameterized constructor takes arguments and is used to initialize objects with specific values.

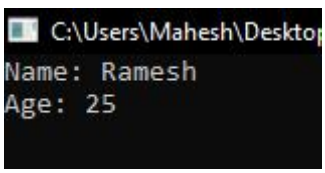
3. Copy Constructor
   A copy constructor initializes an object using another object of the same class. It takes a reference to an object of the same class as a parameter.

PROGRAMS:

a) Write a class Person with a constructor that initializes the name and age of the person.

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

class Persion{
        string name;
        int age;

        public:
                Persion(string n,int a):name(n),age(a){}

                void display(){
                        cout<<"Name: "<<name<<endl;
                        cout<<"Age: "<<age<<endl;
                }
};
int main(){
        Persion p("Ramesh",25);
        p.display();

        return 0;
}
```

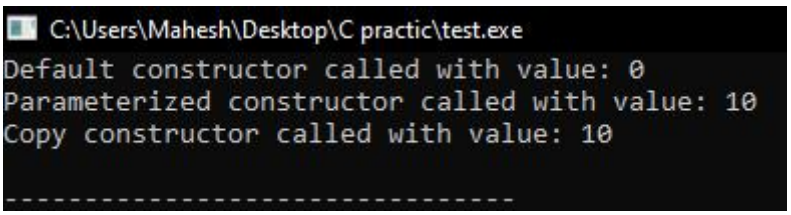
Output:



b) Write a program to demonstrate the use of different types of constructors in C++.

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

class Example {
        public:
            int x;

            Example() {     // Default constructor
                x = 0;
                cout << "Default constructor called with value: " << x << endl;
            }
            Example(int a) {    // Parameterized constructor
                x = a;
                cout << "Parameterized constructor called with value: " << x << endl;
            }
            Example(const Example &obj) {     // Copy constructor
                x = obj.x;
                cout << "Copy constructor called with value: " << x << endl;
            }
};

int main() {
    Example ex1; // Using default constructor

    Example ex2(10); // Using parameterized constructor

    Example ex3 = ex2;     // Using copy constructor

    return 0;
}
```

Output:



CONCLUSION :

A constructor in C++ is to initialize an object's data members and allocate resources when the object is created. It ensures that the object is in a valid state from the moment it is instantiated. In this lab work, we have understod the different types of constructor in C++ programming. Students have get well knowledge of concept of constructor in C++ programming language.