WEEK 10

Program No:10.1

Develop a C++ program that demonstrates the use of function templates to create functions that can work with different data types.

Aim: Develop a C++ program that demonstrates the use of function templates to create functions that can work with different data types.

Description:

Function templates in C++ allow you to write generic functions that work with any data type. This promotes code reusability and type safety by letting the compiler generate the appropriate function based on the type used during the call.

Syntax:

```
template <typename T>
void functionName(T arg1, T arg2) {
   // Function body using type T
}
```

Program:

```
    #include<iostream>
    using namespace std;
    template<typename T>
    T mymax(T x, T y) {
       return (x > y) ? x : y;
    }
    template<typename T>
void swapValues(T &a,T&b) {
       T temp = a;
       a = b;
       b = temp;
    }
    int main() {
  cout<<"Rollno:24B11AI389"<<endl;
  cout << "Max of 4 and 7: " << mymax<int>(4, 7) << endl;
  cout << "Max of 4.0 and 7.0: " << mymax<double>(4.0, 7.0) << endl;
  cout << "Max of 'g' and 'd': " << mymax<char>('g', 'd') << endl;
  int a = 10, b = 30;
  cout << "\nBefore swap : a = " << a << ", b = " << b << endl;
       swapValues(a, b);
cout << "After swap : a = " << a << ", b = " << b << endl;
      double c = 5.0, d = 20.0;
 cout << "\nBefore swap : c = " << c << ", d = " << d << endl;
   swapValues(c, d);

   cout << "After swap : c = " << c << ", d = " << d << endl;
          char e = 'A', f = 'C';

    cout << "\nBefore swap : e = " << e << ", f = " << f << endl;
          swapValues(e, f);

    cout << "After swap : e = " << e << ", f = " << f << endl;
          return 0;

}
```

Output:

```
Roll no:24B11AI389
Max of 4 and 7: 7
Max of 4.0 and 7.0: 7
Max of 'g' and 'd': g
Before swap : a = 10, b = 30
After swap : a = 30, b = 10
Before swap : c = 5, d = 20
After swap : c = 20, d = 5
Before swap : e = A, f = C
After swap : e = C, f = A
```

Date: **Roll No.:**

Program No :10.2

Develop a C++ program that demonstrates template classes, which allow creating classes that can work with any data type

Aim : To develop a C++ program that demonstrates template classes, which allow creating classes that can work with any data type.

Description :

This C++ program uses template classes to create a generic Box class that can store and retrieve values of any data type—like int, double, or string. By using template , the class becomes flexible and reusable, eliminating the need to write separate classes for each type. It demonstrates type safety, code reuse, and the power of generic programming in C++.

Syntax :

```
template<class T>
class ClassName {
private:
      T data;
public:
    ClassName(T value)
{
data = value;
}
void display()
{
cout << "Data: " << data << endl;
}
};
// Creating objects for different data types
ClassName obj1(10);
ClassName obj2(5.5);
```

Program:

```
#include<iostream>
using namespace std;
template <class T>
class calculator
{
        private:
                T num1;
                T num2;
                public:
        calculator(T n1,T n2)
{
```

```
                num1=n1;

                num2=n2;

}
T add()
{
        return num1+num2;
}
T subtract()
{
        return num1-num2;
}
T multiply()
{
        return num1*num2;
}
T divide()
{
        if(num2!=0)
        return num1/num2;
        else
        {
                cout<<"Error!Division by zero."<<endl;
        }
}
};
int main()
{
        cout<<"Roll no:24B11AI389"<<endl;
        calculator<int> intcalc(10,2);
        cout<<"Int calculation:"<<endl;
        cout<<"Addition = "<<intcalc.add()<<endl;
        cout<<"subtraction = "<<intcalc.subtract()<<endl;
        cout<<"Multiplication = "<<intcalc.multiply()<<endl;
        cout<<"Division = "<<intcalc.divide()<<endl;
        cout<<"------------------------"<<endl;

        calculator<double> doublecalc(11.5,5.5);
        cout<<"double calculation:"<<endl;
        cout<<"Addition = "<<doublecalc.add()<<endl;
        cout<<"subtraction = "<<doublecalc.subtract()<<endl;
        cout<<"Multiplication = "<<doublecalc.multiply()<<endl;
        cout<<"Division = "<<doublecalc.divide()<<endl;
        return 0;
}
```

Output:

```
Roll no:24B11AI389
Int calculation:
Addition = 12
subtraction = 8
Multiplication = 20
```

```
Division = 5
-------------------------
double calculation:
Addition = 17
subtraction = 6
Multiplication = 63.25
Division = 2.09091
```