EXPERIMENT-11

OBJECTIVE

Write programs to understand function template and class template.

PROGRAM

Templates: A C++ template is a powerful feature added to C++. It allows you to define the generic classes and generic functions and thus provides support for generic programming. Generic programming is a technique where generic types are used as parameters in algorithms so that they can work for a variety of data types.

Templates can be represented in two ways:

- Function templates
- Class templates

Function Templates:

- We can define a template for a function. For example, if we have an add() function, we can create versions of the add function for adding the int, float or double type values.
- Generic functions use the concept of a function template. Generic functions define a set of operations that can be applied to the various types of data.
- The type of the data that the function will operate on depends on the type of the data passed as a parameter.
- For example, Quick sorting algorithm is implemented using a generic function, it can be implemented to an array of integers or array of floats.
- A Generic function is created by using the keyword template. The template defines what function will do.

Syntax:

```
1. template < class Ttype> ret_type func_name(parameter_list)
2. {
3.     // body of function.
4. }
```

Where

Ttype: It is a placeholder name for a data type used by the function. It is used within the function

definition. It is only a placeholder that the compiler will automatically replace this placeholder with the actual data type.

class: A class keyword is used to specify a generic type in a template declaration.

Program:

```
1.  #include <iostream>
2.  using namespace std;
3.  template<class T> T add(T &a,T &b)
4.  {
5.      T result = a+b;
6.      return result;
7.
8.  }
9.  int main()
10. {
11.   int i =2;
12.   int j =3;
13.   float m = 2.3;
14.   float n = 1.2;
15.   cout<<"Addition of i and j is :"<<add(i,j);
16.   cout<<'\n';
17.   cout<<"Addition of m and n is :"<<add(m,n);
18.   return 0;
19. }
```

Output:

Addition of i and j is :5

Addition of m and n is :3.5

Class Template:

Class Template can also be defined similarly to the Function Template. When a class uses the concept of Template, then the class is known as generic class.

Syntax:

1. **template**<**class** Ttype>

```
2. class class_name
3. {
4.   .
5.   .
6. }
```

Ttype is a placeholder name which will be determined when the class is instantiated. We can define more than one generic data type using a comma-separated list. The Ttype can be used inside the class body.

Now, we create an instance of a class.

```
1. class_name<type> ob;
```

where class_name: It is the name of the class.

type: It is the type of the data that the class is operating on.

ob: It is the name of the object.

Program:

```
1. #include <iostream>
2. using namespace std;
3. template<class T>
4. class A
5. {
6.     public:
7.     T num1 = 5;
8.     T num2 = 6;
9.     void add()
10.    {
11.        std::cout << "Addition of num1 and num2 : " << num1+num2<<std::endl;
12.    }
13.
14. };
15.
16. int main()
17. {
18.    A<int> d;
19.    d.add();
```

```
20.     return 0;
21. }
```

Output:

Addition of num1 and num 2 : 11