

Institute of Computer Technology  
B. Tech Computer Science and Engineering  
Subject: ESFP-II (2CSE203)

**PRACTICAL-6**

**AIM: - To learn about operator overloading & function overloading in C++.**

**1. Create a class that contains a single private char. Overload the iostream operators << and >> and test them. You can test them with string streams, and cin and cout.**

**CODE:**

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

class Employee{
    char name[20];
public:
    friend void operator>>(istream &prajapati,Employee &ob){
        cout<<"Enter Employee Name=";
        prajapati>>ob.name;
    }
    friend void operator<<(ostream &yash,Employee &ob1){
        yash<<"\nYour name is= "<<ob1.name;
    }
};

int main(){
    Employee obj;
    cin>>obj;
    cout<<obj;
    return 0;
}
```

**OUTPUT:**

```
Enter Employee Name=YashPrajapati
```

```
Your name is= YashPrajapati
```

```
PS C:\Users\Admin\Google Drive\B-Tech\SEM-2\ESFP-2\ESFP-Practicals\Prac-6> █
```

**2. Is it possible to apply Overloading mechanism on binary minus operator - using friend function.? If yes, then write suitable code.**

**CODE:**

```
#include<iostream>
using namespace std;
class MinusOP
{
```

```

public:
int x;
void showdata(int a)
{
    x=a;
}
void display()
{
    cout<<"Value of x: "<<x;
}
friend void operator -(MinusOP &obj)
{
    obj.x=-obj.x;
    cout<<"\nValue of x: "<<obj.x;
}
};
int main()
{
    MinusOP obj1;
    obj1.showdata(10);
    obj1.display();
    operator -(obj1);
    return 0;
}

```

**OUTPUT:**

```

Value of x: 10
Value of x: -10
PS C:\Users\Admin\Google Drive\B-Tech\SEM-2\ESFP-2\ESFP-Practicals\Prac-6>

```

**3. Implement following functions in single C++ Program:****void display();****void display( int );****void display( float );****void display( int, float );****CODE:**

```

#include<iostream>
using namespace std;
class OverLoading
{
public:
void display()
{
    cout<<"Yash Prajapati";
}
}

```

```

void display(int a)
{
    cout<<"\nAddition="<<(a+a);
}
void display(float b)
{
    cout<<"\nMultiplication="<<(b*b);
}
void display(int a,float b)
{
    cout<<"\nDivision="<<(a/b);
}
};
int main()
{
    OverLoading obj;
    obj.display();
    obj.display(20);
    obj.display(50.1f);
    obj.display(15,6.1f);
    obj.display(90,10);
    return 0;
}

```

**OUTPUT:**

```

Yash Prajapati
Addition=40
Multiplication=2510.01
Division=2.45902
Division=9
PS C:\Users\Admin\Google Drive\B-Tech\SEM-2\ESFP-2\ESFP-Practicals\Prac-6>

```

**Post Practical Task**

**1. Create two classes, Apple and Orange. In Apple, create a constructor that takes an Orange as an argument. Create a function that takes an Apple and call that function with an Orange to show that it works. Now make the Apple constructor explicit to demonstrate that the automatic type conversion is thus prevented. Modify the call to your function so that the conversion is made explicitly and thus succeeds.**

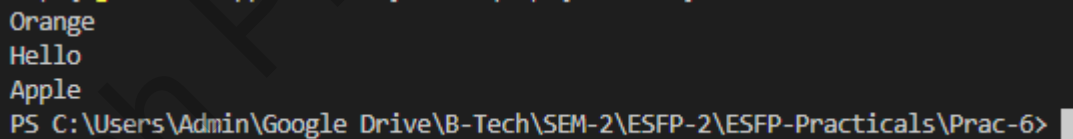
**CODE:**

```

#include <iostream>
using namespace std;
class Apple
{
    public:

```

```
void print()
{
    cout<<"Apple";
}
};
class Orange
{
public:
    Orange()
    {
        cout<<"Orange\n";
    }
    void abc(Apple a)
    {
        cout<<"Hello\n";
        a.print();
    }
    void Display()
    {
        cout<<"Display Method";
    }
};
int main()
{
    Orange b;
    Apple a;
    b.abc(a);
    return 0;
}
```

**OUTPUT:**

```
Orange
Hello
Apple
PS C:\Users\Admin\Google Drive\B-Tech\SEM-2\ESFP-2\ESFP-Practicals\Prac-6>
```

```
2. #include <iostream>
using namespace std;
int fun()
{
    return 1;
}
float fun()
```

```

{
return 10.23;
}
void main()
{
cout <<(int)fun() << ' ';
cout << (float)fun() << ' ';
}

```

**ERROR:**

Cannot overload functions distinguished by return type alone.

**CORRECTION:**

```

#include<iostream>
using namespace std;
int fun()
{
    return 1;
}
float fun1()
{
    return 10.23;
}

int main()
{
    cout<<fun()<<" ";
    cout<<fun1()<<" ";
}

```

**OUTPUT:**

```

1 10.23
PS C:\Users\Admin\Google Drive\B-Tech\SEM-2\ESFP-2\ESFP-Practicals\Prac-6>

```

```

3. #include <iostream>
using namespace std;
int gValue=10;
void extra()
{
cout << gValue << ' ';
}
int main()
{
extra();
{
int gValue = 20;

```

```
cout << gValue << ' ';
cout << gValue << ' ';
}
}
```

# OUTPUT:

```
10 20 20
PS C:\Users\Admin\Google Drive\B-Tech\SEM-2\ESFP-2\ESFP-Practicals\Prac-6> 
```

Yash Prajapati (20162121023)