lab-5

5. Create a Distance class with:
• Feet and inches as data members
• Member function to input distance
• Member function to output distance
• Member function to add two distance objects
a).Write a main function to create objects of DISTANCE class. Input two distances and output the sum.
SourceCode:

```
#include <iostream>
using namespace std;
class Distance
{
	private:
		int feet;
		int inches;
	public:
		void set_distance()
		{
			cout<<"Enter feet: ";
			cin>>feet;
			cout<<"Enter inches: ";
			cin>>inches;
		}
		void get_distance()
		{
			cout<<"Distance is feet= "<<feet<<", inches= "<<inches<<endl;
		}
		void add(Distance d1, Distance d2)
		{
			feet = d1.feet + d2.feet;
			inches = d1.inches + d2.inches;
			feet = feet + (inches / 12);
			inches = inches % 12;
		}
};
int main()
{
	Distance d1, d2, d3;
	d1.set_distance();
	d2.set_distance();
	d3.add(d1, d2);
	d3.get_distance();
	return 0;
}
```

Output:

```
Enter feet: 3
Enter inches: 8
Enter feet: 4
Enter inches: 9
Distance is feet= 8, inches= 5
```

b) Write a C++ Program to illustrate the use of Constructors and Destructors

Sourcecode:

```
#include <iostream>
using namespace std;
class Distance
{
	private:
		int feet;
		int inches;
	public:
		Distance() {}
		Distance(int f, int i)
		{
			feet = f;
			inches = i;
		}
		void get_distance()
		{
			cout<<"Distance is feet= "<<feet<<", inches= "<<inches<<endl;
		}
		void add(Distance &d1, Distance &d2)
		{
			feet = d1.feet + d2.feet;
			inches = d1.inches + d2.inches;
			feet = feet + (inches / 12);
			inches = inches % 12;
		}
		~Distance()
		{
			cout<<"Distance object destroyed"<<endl;
		}
};
int main()
{
	int f1, in1, f2, in2;
	cout<<"Enter feet: ";
	cin>>f1;
	cout<<"Enter inches: ";
	cin>>in1;
	cout<<"Enter feet: ";
	cin>>f2;
	cout<<"Enter inches: ";
	cin>>in2;
	Distance d1(f1, in1);
	Distance d2(f2, in2);
```

```
	Distance d3;
	d3.add(d1, d2);
	d3.get_distance();
	return 0;
}
```

Output:

```
Enter feet: 3
Enter inches: 8
Enter feet: 4
Enter inches: 9
Distance is feet= 8, inches= 5
Distance object destroyed
Distance object destroyed
Distance object destroyed
```

c) Write a program for illustrating function overloading in adding the distance between objects

Sourcecode:

```
#include <iostream>
using namespace std;
class Distance
{
	private:
		int feet;
		int inches;
	public:
		void set_distance()
		{
			cout<<"Enter feet: ";
			cin>>feet;
			cout<<"Enter inches: ";
			cin>>inches;
		}
		void get_distance()
		{
			cout<<"Distance is feet= "<<feet<<", inches= "<<inches<<endl;
		}
		void add(Distance d1, Distance d2)
		{
			feet = d1.feet + d2.feet;
			inches = d1.inches + d2.inches;
			feet = feet + (inches / 12);
			inches = inches % 12;
		}
		void add(Distance *d1, Distance *d2)
		{
			feet = d1->feet + d2->feet;
			inches = d1->inches + d2->inches;
			feet = feet + (inches / 12);
```

```
                inches = inches % 12;
        }
};
int main()
{
        Distance d1, d2, d3;
        d1.set_distance();
        d2.set_distance();
        d3.add(d1, d2);
        d3.get_distance();
        d3.add(&d1, &d2);
        d3.get_distance();
        return 0;
}
```

OutPut:

```
Enter feet: 3
Enter inches: 4
Enter feet: 4
Enter inches: 9
Distance is feet= 8, inches= 1
Distance is feet= 8, inches= 1
```

d) Write a C++ program demonstrating a Bank Account with necessary methods and variables.

Source code:

```
#include <iostream>
using namespace std;
class Account
{
        private:
                string accno;
                string name;
                float balance;
                string bname;
                string brname;
                string ifsc;
        public:
                Account(string acc, string n)
                {
                        accno = acc;
                        name = n;
                        balance = 500;
                        bname = "SBI";
                        brname = "BVRM MAIN";
                        ifsc = "SBI0000818";
                }
                void withdraw(float amount)
                {
```

```
                balance = balance - amount;
                cout<<"Amount withdrawn = "<<amount<<endl;
                cout<<"Remaining balance = "<<balance<<endl;
        }
        void deposit(float amount)
        {
                balance = balance + amount;
                cout<<"Amount deposited = "<<amount<<endl;
                cout<<"Remaining balance = "<<balance<<endl;
        }
        float get_balance()
        {
                return balance;
        }
};
int main()
{
        string acc, n;
        cout<<"Enter account no: ";
        cin>>acc;
        cout<<"Enter your name: ";
        cin>>n;
        Account a1(acc, n);
        a1.deposit(5000);
        a1.withdraw(2200);
        cout<<"Available balance is: "<<a1.get_balance();
        return 0;
}
```

OutPut:

```
Enter account no: 23458
Enter your name: teja
Amount deposited = 5000
Remaining balance = 5500
Amount withdrawn = 2200
Remaining balance = 3300
Available balance is: 3300
```

lab-6

6. Write a program for illustrating Access Specifiers public, private, protected

6(a) A C++ program to illustrate pointer to a class

Source Code:

```
#include <iostream>
using namespace std;
class A
{
	private:
		int x;
		int y;
	public:
		A(int x, int y)
		{
			this->x = x;
			this->y = y;
		}
		void display()
		{
			cout<<"x = "<<x<<endl;
			cout<<"y = "<<y<<endl;
		}
};
int main()
{
	A *ptr = new A(10, 30); //Here ptr is pointer to class A
	ptr->display();
	return 0;
}
Output:
	x = 10
	y = 30
```

6(b) A C++ program to illustrate this pointer.

Source Code:

```
#include <iostream>
using namespace std;
class A
{
	private:
		int x;
		int y;
	public:
		A(int x, int y)
		{
```

```
                this->x = x;
                this->y = y;
        }
        void display()
        {
                cout<<"x = "<<x<<endl;
                cout<<"y = "<<y<<endl;
        }
        A& clone()
        {
                return *this;
        }
};
int main()
{
        A obj1(10, 20);
        obj1.display();
        A obj2 = obj1.clone();
        obj2.display();
        return 0;
 }
```

OutPut:

```
x = 10
y = 20
x = 10
y = 20
```

6(c) A C++ program for implementing friend function.

Source code:

```
#include <iostream>
using namespace std;
class A
{
        private:
                int x;
        public:
                A(int p)
                {
                        x = p;
                }
                friend void display(A &);
};
void display(A &obj)
{
        cout<<"x = "<<obj.x;
}
int main()
```

```
{
        A obj(10);
        display(obj);
        return 0;
}
```

Output:

x = 10

lab-7

7. Write a program to Overload Unary, and Binary Operators as Member Function, and Non Member Function.

a) Unary operator as member function

```
#include <iostream>
using namespace std;
class Number
{
    private:
        int x;
    public:
        Number(int p)
        {
            x = p;
        }
        void operator -()
        {
            x = -x;
        }
        void display()
        {
            cout<<"x = "<<x;
        }
};
int main()
{
    Number n(10);
    -n;
    n.display();
    return 0;
}
```

Output:

x = -10

b) Binary operator as non member function

Sourcecode:

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

```
class Complex
{
	private:
		float real;
		float imag;
	public:
		Complex(){}
		Complex(float r, float i)
		{
			real = r;
			imag = i;
		}
		void display()
		{
			cout<<real<<"+i"<<imag;
		}
		friend Complex operator +(Complex &, Complex &);
};
Complex operator +(Complex &c1, Complex &c2)
{
	Complex temp;
	temp.real = c1.real + c2.real;
	temp.imag = c1.imag + c2.imag;
	return temp;
}
int main()
{
	Complex c1(3, 4);
	Complex c2(4, 6);
	Complex c3 = c1+c2;
	c3.display();
	return 0;
}
```

Output:

7+i10

c) Write a c ++ program to implement the overloading assignment = operator

Sourcecode:

```
#include <iostream>
using namespace std;
class Number
{
	private:
		int x;
	public:
		Number(int p)
		{
			x = p;
		}
		Number operator =(Number &n)
		{
```

```
                return Number(n.x);
        }
        void display()
        {
                cout<<"x = "<<x;
        }
};
int main()
{
        Number n1(10);
        Number n2 = n1;
        n2.display();
        return 0;
}
```

Output:

```
x = 10
```