Overriding, overloading, constructor in inheritance

1. Create a class FLOAT that contains one float data member. Overload all the four arithmetic operators so that they can operate on the objects of FLOAT

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

class FLOAT
{
private:
    float a;

public:
    FLOAT()
    {
        a = 0;
    }
    FLOAT(float x)
    {
        a = x;
    }
    float operator+(FLOAT X)
    {
        return (a + X.a);
    }
    float operator-(FLOAT X)
    {
        return (a - X.a);
    }
    float operator*(FLOAT X)
    {
        return (a * X.a);
    }
    float operator/(FLOAT X)
    {
        return (a / X.a);
    }
};

int main()
{
    FLOAT f1(10), f2(2);
    float result;
    result = f1 + f2;
    cout << "Sum is " << result << endl;
    result = f1 - f2;
    cout << "Subtraction is " << result << endl;
    result = f1 * f2;
    cout << "Product is " << result << endl;
    result = f1 / f2;
    cout << "Division is " << result << endl;
    return 0;
}
```

```
Output;
Sum is 12
Subtraction is 8
Product is 20
Division is 5
```

.

2. Define a class Rectangle and overload area function for different types of data type.

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

class Rectangle
{
public:
    void area(int x, int y)
    {
        cout << "Area of rectangle is " << x * y << " Unit" << endl;
    }
    void area(double x, double y)
    {
        cout << "Area of rectangle is " << x * y << " Unit" << endl;
    }
    void area(int x, double y)
    {
        cout << "Area of rectangle is " << x * y << " Unit" << endl;
    }
    void area(double x, int y)
    {
        cout << "Area of rectangle is " << x * y << " Unit" << endl;
    }
};

int main()
{
    Rectangle r, r2;
    r.area(10, 10);
    r.area(14.28, 13.9);
    r2.area(2.14, 45);
    return 0;
}
```

```
Output:
Area of rectangle is 100 Unit
Area of rectangle is 198.492 Unit
Area of rectangle is 96.3 Unit
```

3. Define a base class Animals having member function sound() . Define another derived class from Animals class named Dogs. You need to override the sound function of the base class in the derived class.

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

class Animal
{
public:
    virtual void sound()
    {
        cout << "Animal class " << endl;
    }
};
class Dogs : public Animal
{
public:
    void sound()
    {
        cout << "Dog class" << endl;
    }
};

int main()
{
```

```
    Animal *a, a1;
    Dogs d1;
    a = &d1;
    a->sound();
    a = &a1;
    a->sound();
    return 0;
}
================================================================================
Output:
Dog class
Animal class
```

4. Define a class Addition that can add 2 or 3 numbers of different data types using function overloading.

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

class Addition
{
    public:
        void add(int x, int y, int z = 0)
        {
             cout << x  << " + "<< y << " + " << z  << " = " << x + y + z <<
endl;
        }
        void add(double x, double y, double z = 0)
        {
             cout << x  << " + "<< y << " + " << z  << " = " << x + y + z <<
endl;
        }
        void add(int x, double y, double z = 0)
        {
             cout << x  << " + "<< y << " + " << z  << " = " << x + y + z <<
endl;
        }
        void add(int x, double y, int z = 0)
        {
             cout << x  << " + "<< y << " + " << z  << " = " << x + y + z <<
endl;
        }
        void add(double x, int y, int z = 0)
        {
             cout << x  << " + "<< y << " + " << z  << " = " << x + y + z <<
endl;
        }
        void add(double x, int y, double z = 0)
        {
             cout << x  << " + "<< y << " + " << z  << " = " << x + y + z <<
endl;
        }
};

int main()
{
    Addition a1, a2, a3;
    a1.add(5, 10, 15);
    a2.add(5.25, 12.254, 145.234);
    a3.add(568, 254.321, 2.3);
    return 0;
}
================================================================================
Output:
```

```
5 + 10 + 15 = 30
5.25 + 12.254 + 145.234 = 162.738
568 + 254.321 + 2.3 = 824.621
```

5. Define a class A having multiple constructors. Define another class B derived from class A. Create derived class constructors and show use of constructor in this single inheritance.

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

class A
{
public:
    A()
    {
        cout << "Base class Without argument constructor" << endl;
    }
    A(int x)
    {
        cout << "Base class 1-Argument constructor" << endl;
    }
};
class B : public A
{
public:
    B():A()
    {
        cout << "Derived class Without argument constructor" << endl;
    }
    B(int x):A(8)
    {
        cout << "Derived class 1-Argument constructor" << endl;
    }
};
int main()
{
    B b1, b2(5);
    return 0;
}
================================================================================
Output:
Base class Without argument constructor
Derived class Without argument constructor
Base class 1-Argument constructor
Derived class 1-Argument constructor
```

6. C++ Program to illustrate the use of Constructors in multilevel inheritance of your choice.

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

class Animal
{
public:
    void eat()
    {
        cout << "Eating..." << endl;
    }
};
class Dog : public Animal
{
public:
    void bark()
    {
        cout << "Barking..." << endl;
```

```
    }
};
class Puppy : public Dog
{
public:
    void sleep()
    {
        cout << "Sleeping..." << endl;
    }
};

int main()
{
    Puppy p1;
    p1.bark();
    p1.eat();
    p1.sleep();
    return 0;
}
```

```
Output:
Barking...
Eating...
Sleeping...
```

7. C++ Program to illustrate the use of Constructors in single inheritance of your choice.

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

class Addition
{
private:
    float a, b, result;

public:
    Addition()
    {
        cout << "Enter two numbers: ";
        cin >> a >> b;
        result = a + b;
    }
    void display()
    {
        cout << "Sum is " << result << endl;
    }
};

class Sum : public Addition
{
public:
    Sum() : Addition()
    {
    }
};

int main()
{
    Sum s;
    s.display();
    return 0;
}
```

```
Output:
Enter two numbers: 20 30
```

```
Sum is 50
```

8. Write a C++ program to find the factorial of a number using copy constructor

```
#include <iostream>
using namespace std;
class Factorial
{
private:
    int fact, number, i;

public:
    Factorial() {}
    Factorial(int x)
    {
        fact = 1;
        number = x;
    }
    Factorial(Factorial &X)
    {
        number = X.number;
        fact = 1;
    }
    void calculate()
    {
        for (i = 1; i <= number; i++)
        {
            fact = fact * i;
        }
        }
    void display()
    {
        cout << "Factorial is " << fact << endl;
    }
};
int main()
{
    Factorial f1(5); // normal parameterised constructor called
    f1.calculate();
    f1.display();

    Factorial f2(f1); // copy constructor called
    f2.calculate();
    f2.display();
    return 0;
}
================================================================================
Output:
Factorial is 120
Factorial is 120
```

9. Write a C++ program to calculate the area of triangle, rectangle and circle using constructor overloading. The program should be menu driven.

```
#include <iostream>
#include <math.h>
using namespace std;

class Area
{
private:
    float area;

public:
    Area(float R)
```

```
    {
        area = (3.14159 * R * R);
    }
    Area(float A, float B, float C)
    {
        float S;
        S = (A + B + C) / 2;
        area = (S * (S - A) * (S - B) * (S - C));
        area = pow(area, 0.5);
    }
    Area(float L, float B)
    {
        area = L * B;
    }
    void display()
    {
        cout << "Area is " << area << endl;
    }
};

int main()
{
    float a, b, c;
    int ch;
    while (1)
    {
        cout << "1. Area of circle" << endl;
        cout << "2. Area of rectangle" << endl;
        cout << "3. Area of triangle" << endl;
        cout << "4. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> ch;

        switch (ch)
        {
        case 1:
        {
            cout << "Enter radius of a circle: ";
            cin >> a;
            Area a1(a);
            a1.display();
            break;
        }
        case 2:
        {
            cout << "Enter length and breadth of rectangle: ";
            cin >> a >> b;
            Area a2(a, b);
            a2.display();
            break;
        }
        case 3:
        {
            cout << "Enter three sides of triangle: ";
            cin >> a >> b >> c;
            Area a3(a, b, c);
            a3.display();
            break;
        }
        case 4:
            exit(0);
        default:
            cout << "Invalid Input" << endl;
            break;
```

```
        }
    }

    return 0;
}
```

```
Output:
1. Area of circle
2. Area of rectangle
3. Area of triangle
4. Exit
Enter your choice: 1
Enter radius of a circle: 15.2
Area is 725.833
1. Area of circle
2. Area of rectangle
3. Area of triangle
4. Exit
Enter your choice: 2
Enter length and breadth of rectangle: 10 40
Area is 400
1. Area of circle
2. Area of rectangle
3. Area of triangle
4. Exit
Enter your choice: 3
Enter three sides of triangle: 3 6 7
Area is 8.94427
1. Area of circle
2. Area of rectangle
3. Area of triangle
4. Exit
Enter your choice: 4
```

10. Create a C++ class for player objects with the following attributes: player no., name, number of matches and number of goals done in each match. The number of matches varies for each player. Write a parameterized constructor which initializes player no., name, number of subjects and creates an array for number of goals and number of matches dynamically.

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

class Player
{
private:
    int player_no, number_of_matches, *number_of_goals;
    char name[50];

public:
    Player()
    {
        cout << "Enter player no.: ";
        cin >> player_no;
        cout << "Enter player name: ";
        cin.ignore();
        cin.getline(name, 50);
        cout << "Enter number of matches: ";
        cin >> number_of_matches;
        number_of_goals = new int[number_of_matches];
        for (int i = 0; i <= number_of_matches - 1; i++)
        {
            cout << "Enter number of golas in match " << i + 1 << ": ";
            cin >> number_of_goals[i];
```

```
        }
    }
    void display()
    {
        cout << "\n--------------------------------" << endl;
        cout << "Player No. : " << player_no << endl;
        cout << "Player Name : " << name << endl;
        cout << "No. of Matches Played : " << number_of_matches << endl;
        for (int i = 0; i <= number_of_matches - 1; i++)
            cout << "Match " << i + 1 << " Goals : " << number_of_goals[i] <<
endl;
    }
};

int main()
{
    Player p1, p2;
    p1.display();
    p2.display();
    return 0;
}
```

==========

```
Output:
Enter player no.: 1
Enter player name: Shekh Akhtar
Enter number of matches: 3
Enter number of golas in match 1: 5
Enter number of golas in match 2: 4
Enter number of golas in match 3: 4
Enter player no.: 2
Enter player name: Mukesh Rathore
Enter number of matches: 2
Enter number of golas in match 1: 4
Enter number of golas in match 2: 4

--------------------------------
Player No. : 1
Player Name : Shekh Akhtar
No. of Matches Played : 3
Match 1 Goals : 5
Match 2 Goals : 4
Match 3 Goals : 4

--------------------------------
Player No. : 2
Player Name : Mukesh Rathore
No. of Matches Played : 2
Match 1 Goals : 4
Match 2 Goals : 4
```