Practice questions on Classes and objects

Create a class named 'Student' with a string variable 'name' and an integer variable 'roll_no'. Assign the value of roll_no as '2' and that of name as "John" by creating an object of the class Student.

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

class Student
{
    public:
        string name;
        int roll_no;

};

int main()
{
    Student s;
    s.name = "John";
    s.roll_no = 2;
    cout << s.name << " " << s.roll_no << endl;
    return 0;
}
```

Assign and print the roll number, phone number and address of two students having names "Sam" and "John" respectively by creating two objects of the class 'Student'.

#include <iostream>

#include <string>

class Student {

private:

    std::string name;

    int rollNumber;

    std::string phoneNumber;

    std::string address;

```
public:
    Student(const std::string& n, int roll, const std::string& phone, const std::string& addr)
        : name(n), rollNumber(roll), phoneNumber(phone), address(addr) {}

    void printDetails() const {
        std::cout << "Name: " << name << std::endl;
        std::cout << "Roll Number: " << rollNumber << std::endl;
        std::cout << "Phone Number: " << phoneNumber << std::endl;
        std::cout << "Address: " << address << std::endl;
    }
};

int main() {
    Student sam("Sam", 101, "1234567890", "123 Main Street");
    Student john("John", 102, "9876543210", "456 Elm Avenue");

    std::cout << "Details of Student Sam:" << std::endl;
    sam.printDetails();

    std::cout << "\nDetails of Student John:" << std::endl;
    john.printDetails();

    return 0;
}
```

Write a program to print the area and perimeter of a triangle having sides of 3, 4 and 5 units by creating a class named 'Triangle' with a function to print the area and perimeter.

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

class Triangle
{
public:
    void print_area(int s1, int s2, int s3)
    {
        double s = (s1+s2+s3)/2.0;
        cout << s << endl;
        cout << "Perimeter is " << (s1+s2+s3) << endl;
    }
};

int main()
{
    Triangle t;
    t.print_area(3,4,5);
    return 0;
}
```

Write a program to print the area and perimeter of a triangle having sides of 3, 4 and 5 units by creating a class named 'Triangle' with the constructor having the three sides as its parameters.

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

class Triangle
{
public:
    int s1,s2,s3;
    Triangle(int a,int b,int c)
    {
        s1 = a;
        s2 = b;
        s3 = c;
    }
    void print_area()
    {
        double s = (s1+s2+s3)/2.0;
        cout << s << endl;
        cout << "Perimeter is " << (s1+s2+s3) << endl;
    }
};

int main(){
    Triangle t(3,4,5);
```

```
    t.print_area();
    return 0;
}
```

Write a program to print the area of two rectangles having sides (4,5) and (5,8) respectively by creating a class named 'Rectangle' with a function named 'Area' which returns the area. Length and breadth are passed as parameters to its constructor.

```
#include <iostream>


class Rectangle {

private:

   int length;

   int breadth;


public:

   Rectangle(int l, int b) : length(l), breadth(b) {}


   int Area() const {

      return length * breadth;

   }

};


int main() {

   Rectangle rectangle1(4, 5);

   Rectangle rectangle2(5, 8);


   std::cout << "Area of Rectangle 1: " << rectangle1.Area() << std::endl;

   std::cout << "Area of Rectangle 2: " << rectangle2.Area() << std::endl;


   return 0;

}
```

Write a program to print the area of a rectangle by creating a class named 'Area' having two functions. First function named as 'setDim' takes the length and breadth of the rectangle as parameters and the second function named as 'getArea' returns the area of the rectangle. Length and breadth of the rectangle are entered through keyboard.

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

class Area
{
public:
    int length;
    int breadth;
    void setDim(int l, int b)
    {
        length = l;
        breadth = b;
    }
    int getArea()
    {
        return length*breadth;
    }
};

int main()
{
    Area a;
    a.setDim(4,5);
    cout << a.getArea() << endl;
    return 0;
}
```

Write a program to print the area of a rectangle by creating a class named 'Area' taking the values of its length and breadth as parameters of its constructor and having a function named 'returnArea' which returns the area of the rectangle. Length and breadth of the rectangle are entered through keyboard.

#include <iostream>

class Area {

private:

int length;

int breadth;

public:

Area(int l, int b) : length(l), breadth(b) {}

```
    int returnArea() const {
        return length * breadth;
    }
};

int main() {
    int length, breadth;

    std::cout << "Enter the length of the rectangle: ";
    std::cin >> length;

    std::cout << "Enter the breadth of the rectangle: ";
    std::cin >> breadth;

    Area rectangle(length, breadth);
    int area = rectangle.returnArea();

    std::cout << "Area of the rectangle: " << area << std::endl;

    return 0;
}
```

Print the average of three numbers entered by the user by creating a class named 'Average' having a function to calculate and print the average without creating any object of the Average class.

```
#include <iostream>

class Average {
public:
```

```
    static void calculateAndPrintAverage(double num1, double num2, double num3) {
        double average = (num1 + num2 + num3) / 3.0;
        std::cout << "Average: " << average << std::endl;
    }
};

int main() {
    double num1, num2, num3;

    std::cout << "Enter three numbers: ";
    std::cin >> num1 >> num2 >> num3;

    Average::calculateAndPrintAverage(num1, num2, num3);

    return 0;
}
```

Print the sum, difference and product of two complex numbers by creating a class named 'Complex' with separate functions for each operation whose real and imaginary parts are entered by the user.

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

class Complex
{
private:
    int real;
    int imag;
public:
    Complex(int r, int i)
    {
        real = r;
        imag = i;
    }

    int get_real()
    {
        return real;
    }
    int get_imag()
    {
        return imag;
    }

    void add(Complex c1)
    {
        cout << c1.get_real()+real << "+i" << c1.get_imag()+imag << endl;
    }

    void difference(Complex c1)
    {
        cout << real-c1.get_real() << "+i" << imag-c1.get_imag() << endl;
    }

    void multiply(Complex c1)
    {
        cout << ((real*c1.get_real())-(imag*c1.get_imag())) << "+i" << ((real*c1.get_im
ag())+(imag*c1.get_real())) << endl;
    }
};

int main()
{
    Complex c1(4,5);
    Complex c2(2,3);
    c1.add(c2);
    c1.difference(c2);
    c1.multiply(c2);
    return 0;
}
```

Write a program to print the volume of a box by creating a class named 'Volume' with an initialization list to initialize its length, breadth and height. (just to make you familiar with initialization lists)

```
#include <iostream>

class Volume {
private:
    double length;
    double breadth;
    double height;

public:
    // Constructor with initialization list
    Volume(double l, double b, double h) : length(l), breadth(b), height(h) {}

    // Function to calculate and return the volume of the box
    double getVolume() {
        return length * breadth * height;
    }
};

int main() {
    double length, breadth, height;

    std::cout << "Enter the length of the box: ";
    std::cin >> length;

    std::cout << "Enter the breadth of the box: ";
    std::cin >> breadth;

    std::cout << "Enter the height of the box: ";
```

```
    std::cin >> height;

    // Create a Volume object with initialization list
    Volume box(length, breadth, height);

    // Calculate and print the volume of the box
    std::cout << "Volume of the box: " << box.getVolume() << std::endl;

    return 0;
}
```

Write a program that would print the information (name, year of joining, salary, address) of three employees by creating a class named 'Employee'. The output should be as follows:

Name	Year of joining	Address
Robert	1994	64C- WallsStreat
Sam	2000	68D- WallsStreat
John	1999	26B- WallsStreat

```
#include <iostream>
#include <iomanip>
#include <string>

class Employee {
private:
    std::string name;
    int yearOfJoining;
    double salary;
    std::string address;

public:
    Employee(std::string n, int yoj, double sal, std::string addr)
```

```
        : name(n), yearOfJoining(yoj), salary(sal), address(addr) {}

    void displayInfo() {
        std::cout << std::setw(10) << name;
        std::cout << std::setw(18) << yearOfJoining;
        std::cout << std::setw(20) << address << std::endl;
    }
};

int main() {
    Employee emp1("Robert", 1994, 50000.0, "64C- WallsStreat");
    Employee emp2("Sam", 2000, 60000.0, "68D- WallsStreat");
    Employee emp3("John", 1999, 55000.0, "26B- WallsStreat");

    std::cout << std::setw(10) << "Name";
    std::cout << std::setw(18) << "Year of Joining";
    std::cout << std::setw(20) << "Address" << std::endl;

    emp1.displayInfo();
    emp2.displayInfo();
    emp3.displayInfo();

    return 0;
}
```

Add two distances in inch-feet by creating a class named 'AddDistance'.

```
#include <iostream>

class AddDistance {
private:
```

```
    int feet;
    int inches;

public:
    AddDistance(int ft, int in) : feet(ft), inches(in) {}

    void addDistance(AddDistance d1, AddDistance d2) {
        inches = d1.inches + d2.inches;
        feet = d1.feet + d2.feet + (inches / 12);
        inches = inches % 12;
    }

    void displayDistance() {
        std::cout << "Total Distance: " << feet << " feet " << inches << " inches" << std::endl;
    }
};

int main() {
    AddDistance distance1(5, 8); // 5 feet 8 inches
    AddDistance distance2(3, 10); // 3 feet 10 inches

    AddDistance totalDistance(0, 0);
    totalDistance.addDistance(distance1, distance2);

    totalDistance.displayDistance();

    return 0;
}
```

Write a program by creating an 'Employee' class having the following functions and print the final salary.
1 - 'getInfo()' which takes the salary, number of hours of work per day of employee as parameters
2 - 'AddSal()' which adds $10 to the salary of the employee if it is less than $500.
3 - 'AddWork()' which adds $5 to the salary of the employee if the number of hours of work per day is more than 6 hours.

```
#include <iostream>


class Employee {

private:

   double salary;

   int hoursOfWorkPerDay;


public:

   void getInfo(double s, int hours) {

      salary = s;

      hoursOfWorkPerDay = hours;

   }


   void AddSal() {

      if (salary < 500) {

         salary += 10;

      }

   }


   void AddWork() {

      if (hoursOfWorkPerDay > 6) {

         salary += 5;

      }

   }
```

```
    void displaySalary() {
        std::cout << "Final Salary: $" << salary << std::endl;
    }
};

int main() {
    Employee emp;

    double salary;
    int hoursOfWorkPerDay;

    std::cout << "Enter salary: ";
    std::cin >> salary;

    std::cout << "Enter hours of work per day: ";
    std::cin >> hoursOfWorkPerDay;

    emp.getInfo(salary, hoursOfWorkPerDay);
    emp.AddSal();
    emp.AddWork();
    emp.displaySalary();

    return 0;
}
```

Create a class called 'Matrix' containing constructor that initializes the number of rows and the number of columns of a new Matrix object. The Matrix class has the following information:
1 - number of rows of matrix
2 - number of columns of matrix
3 - elements of matrix (You can use 2D vector)
The Matrix class has functions for each of the following:
1 - get the number of rows
2 - get the number of columns
3 - set the elements of the matrix at a given position (i,j)
4 - adding two matrices.
5 - multiplying the two matrices
You can assume that the dimensions are correct for the multiplication and addition.

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

class Matrix
{
private:
    int row,col;
    vector<vector<int>> matrix;
public:
    Matrix(int r, int c, vector<vector<int>> &m)
    {
        row = r;
        col = c;
        matrix = m;
    }

    int get_row_number()
    {
        return row;
    }

    int get_col_number()
    {
        return col;
    }

    vector<vector<int>> get_vector()
    {
        return matrix;
    }

    void set_element(int i, int j, int e)
    {
        matrix[i][j] = e;
    }

    void display()
    {
        for(int i=0; i<row; i++)
```

```
        {
            for(int j=0; j<col; j++)
            {
                cout << matrix[i][j] << "\t";
            }
            cout << endl;
        }
        cout << endl;
    }

    Matrix add(Matrix m)
    {
        //assuming matrices can be added
        vector<vector<int>> v;
        v.resize(row,vector<int>(col,0));
        for(int i=0;i<row;i++)
        {
            for(int j=0;j<col;j++)
            {
                v[i][j] = matrix[i][j]+m.get_vector()[i][j];
            }
        }
        Matrix n(row,col,v);
        return n;
    }

    Matrix multiply(Matrix m)
    {
        //assuming dimension is correct for multiplication
        vector<vector<int>> v;
        v.resize(row,vector<int>(m.get_col_number(),0));
        for(int i=0; i<row; i++)
        {
            for(int j=0; j<m.get_col_number(); j++)
            {
                for(int k=0; k<col; k++)
                {
                     v[i][j] = v[i][j]+(matrix[i][k]*m.get_vector()[k][j]);
                }
            }
        }
        Matrix n(row,m.get_col_number(),v);
        return n;
    }
};

int main()
{
    vector<vector<int>> m{{1,2,3},{4,5,6},{7,8,9}};
    vector<vector<int>> n{{10,11,12},{13,14,15},{16,17,18}};
    Matrix m1(3,3,m);
    Matrix m2(3,3,n);
    m1.display();
    m2.display();
```

```
    Matrix a = m1.add(m2);
    a.display();
    Matrix b = m1.multiply(m2);
    b.display();
    return 0;
}
```

Practice questions on Array of objects

Write a program to print the name, salary and date of joining of 10 employees in a company. Use array of objects.

```
#include <iostream>
#include <string>

class Employee {
private:
    std::string name;
    double salary;
    std::string dateOfJoining;

public:
    void getInfo() {
        std::cout << "Enter name: ";
        std::cin.ignore();
        std::getline(std::cin, name);

        std::cout << "Enter salary: ";
        std::cin >> salary;

        std::cout << "Enter date of joining: ";
```

```
        std::cin.ignore();
        std::getline(std::cin, dateOfJoining);
    }

    void displayInfo() {
        std::cout << "Name: " << name << std::endl;
        std::cout << "Salary: $" << salary << std::endl;
        std::cout << "Date of Joining: " << dateOfJoining << std::endl;
    }
};

int main() {
    const int numEmployees = 10;
    Employee employees[numEmployees];

    std::cout << "Enter information for 10 employees:\n";
    for (int i = 0; i < numEmployees; i++) {
        std::cout << "Employee " << i + 1 << ":\n";
        employees[i].getInfo();
        std::cout << std::endl;
    }

    std::cout << "Employee Information:\n";
    for (int i = 0; i < numEmployees; i++) {
        std::cout << "Employee " << i + 1 << ":\n";
        employees[i].displayInfo();
        std::cout << std::endl;
    }
```

```
    return 0;
}
```

Write a program to print the roll number and average marks of 8 students in three subjects (each out of 100). The marks are entered by the user and the roll numbers are automatically assigned.

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

class Student {
private:
    static int rollNumber;
    int marks[3];
    double averageMarks;

public:
    Student() {
        rollNumber++;
        for (int i = 0; i < 3; i++) {
            cout << "Enter marks for Subject " << i + 1 << ": ";
            cin >> marks[i];
        }
        calculateAverage();
    }

    void calculateAverage() {
        double total = 0.0;
        for (int i = 0; i < 3; i++) {
            total += marks[i];
        }
```

```
        averageMarks = total / 3;
    }

    void displayInfo() {
        cout << "Roll Number: " << setw(2) << setfill('0') << rollNumber << endl;
        cout << "Average Marks: " << averageMarks << endl;
        cout << endl;
    }
};

int Student::rollNumber = 0;

int main() {
    const int numStudents = 8;
    Student students[numStudents];

    cout << "Student Information:\n";
    for (int i = 0; i < numStudents; i++) {
        students[i].displayInfo();
    }

    return 0;
}
```

Write a program to calculate the average height of all the students of a class. The number of students and their heights are entered by the user.

```
#include <iostream>

using namespace std;


int main() {

   int numStudents;

   cout << "Enter the number of students in the class: ";

   cin >> numStudents;


   double totalHeight = 0.0;

   double height;

   for (int i = 1; i <= numStudents; i++) {

      cout << "Enter the height of student " << i << " in meters: ";

      cin >> height;

      totalHeight += height;

   }


   double averageHeight = totalHeight / numStudents;


   cout << "Average height of all students: " << averageHeight << " meters" << endl;


   return 0;

}
```

Lets create a bank account. Create a class named 'BankAccount' with the following data members
1 - Name of depositor
2 - Address of depositor
3 - Type of account
4 - Balance in account
5 - Number of transactions
Class 'BankAccount' has a function for each of the following
1 - Generate a unique account number for each depositor
For the first depositor, account number will be BA1000, for the second depositor it will be BA1001 and so on
2 - Display information and balance of depositor
3 - Deposit more amount in the balance of any depositor
4 - Withdraw some amount from the balance deposited
5 - Change the address of depositor
After creating the class, do the following operations
1 - Enter the information (name, address, type of account, balance) of the depositors. Number of depositors are to be entered by the user.
2 - Print the information of any depositor.
3 - Add some amount to the account of any depositor and then display the final information of that depositor
4 - Remove some amount from the account of any depositor and then display the final information of that depositor
5 - Change the address of any depositor and then display the final information of that depositor
6 - Randomly repeat these processes for some other bank accounts and after that print the total number of transactions.

```
#include <iostream>

#include <string>

using namespace std;


class BankAccount {

private:

    static int accountNumberCounter;

    int accountNumber;

    string name;

    string address;

    string accountType;

    double balance;

    int numTransactions;
```

```
public:
    BankAccount(string n, string add, string type, double bal)
        : name(n), address(add), accountType(type), balance(bal), numTransactions(0) {
        accountNumber = 1000 + accountNumberCounter;
        accountNumberCounter++;
    }

    void displayInfo() {
        cout << "Account Number: BA" << accountNumber << endl;
        cout << "Name: " << name << endl;
        cout << "Address: " << address << endl;
        cout << "Account Type: " << accountType << endl;
        cout << "Balance: $" << balance << endl;
        cout << "Number of Transactions: " << numTransactions << endl;
        cout << endl;
    }

    void deposit(double amount) {
        balance += amount;
        numTransactions++;
        cout << "$" << amount << " deposited successfully. Current balance: $" << balance <<
endl;
    }

    void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            numTransactions++;
            cout << "$" << amount << " withdrawn successfully. Current balance: $" << balance <<
endl;
```

```
        } else {
            cout << "Insufficient balance." << endl;
        }
    }

    void changeAddress(string newAddress) {
        address = newAddress;
        cout << "Address changed successfully. New address: " << address << endl;
    }
};

int BankAccount::accountNumberCounter = 0;

int main() {
    int numDepositors;
    cout << "Enter the number of depositors: ";
    cin >> numDepositors;

    BankAccount* depositors[numDepositors];

    for (int i = 0; i < numDepositors; i++) {
        string name, address, accountType;
        double balance;

        cout << "\nEnter details for depositor " << i + 1 << ":" << endl;
        cout << "Name: ";
        cin.ignore();
        getline(cin, name);

        cout << "Address: ";
```

```
        getline(cin, address);

        cout << "Account Type: ";
        getline(cin, accountType);

        cout << "Balance: $";
        cin >> balance;

        depositors[i] = new BankAccount(name, address, accountType, balance);
    }

    cout << "\nInformation of a depositor (Enter depositor number between 1 to " <<
numDepositors << "): ";
    int depositorNumber;
    cin >> depositorNumber;
    depositors[depositorNumber - 1]->displayInfo();

    cout << "\nDeposit some amount to depositor " << depositorNumber << "'s account: $";
    double depositAmount;
    cin >> depositAmount;
    depositors[depositorNumber - 1]->deposit(depositAmount);

    cout << "\nWithdraw some amount from depositor " << depositorNumber << "'s account: $";
    double withdrawAmount;
    cin >> withdrawAmount;
    depositors[depositorNumber - 1]->withdraw(withdrawAmount);

    cout << "\nChange address of depositor " << depositorNumber << ": ";
    cin.ignore();
    string newAddress;
```

```
    getline(cin, newAddress);
    depositors[depositorNumber - 1]->changeAddress(newAddress);

    // Free memory for depositors
    for (int i = 0; i < numDepositors; i++) {
        delete depositors[i];
    }

    cout << "\nTotal number of transactions: " << BankAccount::accountNumberCounter << endl;

    return 0;
}
```

Write a program to create a directory that contains the following information.
(a) Name of a person
(b) Address
(c) Telephone Number (if available with STD code)
(d) Mobile Number (if available)
(e) Head of the family

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

class DirectoryEntry {
private:
   string name;
   string address;
   string telephone;
   string mobile;
```

```
    string headOfFamily;

public:
    DirectoryEntry(string n, string addr, string tel, string mob, string head)
        : name(n), address(addr), telephone(tel), mobile(mob), headOfFamily(head) {}

    void displayInfo() {
        cout << "Name: " << name << endl;
        cout << "Address: " << address << endl;
        cout << "Telephone: " << telephone << endl;
        cout << "Mobile: " << mobile << endl;
        cout << "Head of Family: " << headOfFamily << endl;
        cout << endl;
    }
};

int main() {
    vector<DirectoryEntry> directory;

    // Add directory entries (you can add more as needed)
    directory.push_back(DirectoryEntry("John Doe", "123 Main Street, City A", "0123456789",
"9876543210", "Doe Family"));
    directory.push_back(DirectoryEntry("Jane Smith", "456 Park Avenue, City B", "0222222222",
"9998887777", "Smith Family"));
    directory.push_back(DirectoryEntry("Michael Johnson", "789 Street Road, City C", "0333333333",
"1112223333", "Johnson Family"));

    // Display the directory
    cout << "Directory Information:\n";
    for (const auto& entry : directory) {
```

```
        entry.displayInfo();
    }

    return 0;
}
```

Practice questions on Multiple inheritance

Create two classes named Mammals and MarineAnimals. Create another class named BlueWhale which inherits both the above classes. Now, create a function in each of these classes which prints "I am mammal", "I am a marine animal" and "I belong to both the categories: Mammals as well as Marine Animals" respectively. Now, create an object for each of the above class and try calling
1 - function of Mammals by the object of Mammal
2 - function of MarineAnimal by the object of MarineAnimal
3 - function of BlueWhale by the object of BlueWhale
4 - function of each of its parent by the object of BlueWhale

```
#include <iostream>

using namespace std;


// Base class Mammals

class Mammals {

public:

    void printMammal() {

        cout << "I am a mammal." << endl;

    }

};


// Base class MarineAnimals

class MarineAnimals {

public:

    void printMarineAnimal() {
```

```
        cout << "I am a marine animal." << endl;
    }
};

// Derived class BlueWhale inheriting both Mammals and MarineAnimals
class BlueWhale : public Mammals, public MarineAnimals {
public:
    void printBlueWhale() {
        cout << "I belong to both the categories: Mammals as well as Marine Animals." << endl;
    }
};

int main() {
    Mammals mammalObj;
    MarineAnimals marineAnimalObj;
    BlueWhale blueWhaleObj;

    // Function of Mammals using the object of Mammals
    mammalObj.printMammal();

    // Function of MarineAnimals using the object of MarineAnimals
    marineAnimalObj.printMarineAnimal();

    // Function of BlueWhale using the object of BlueWhale
    blueWhaleObj.printBlueWhale();

    // Function of each of its parent classes using the object of BlueWhale
    blueWhaleObj.printMammal();
    blueWhaleObj.printMarineAnimal();
```

```
    return 0;
}
```

Make a class named Fruit with a data member to calculate the number of fruits in a basket. Create two other class named Apples and Mangoes to calculate the number of apples and mangoes in the basket. Print the number of fruits of each type and the total number of fruits in the basket.

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

// Base class Fruit
class Fruit {
protected:
    int totalFruits; // Number of fruits in the basket

public:
    Fruit() : totalFruits(0) {}

    void addToBasket(int numFruits) {
        totalFruits += numFruits;
    }

    int getTotalFruits() {
        return totalFruits;
    }
};

// Subclass Apples
class Apples : public Fruit {
public:
    void addToBasket(int numApples) {
```

```
        totalFruits += numApples;
    }

    int getNumApples() {
        return totalFruits;
    }
};

// Subclass Mangoes
class Mangoes : public Fruit {
public:
    void addToBasket(int numMangoes) {
        totalFruits += numMangoes;
    }

    int getNumMangoes() {
        return totalFruits;
    }
};

int main() {
    Fruit fruitBasket;
    Apples appleBasket;
    Mangoes mangoBasket;

    appleBasket.addToBasket(5); // Add 5 apples to the basket
    mangoBasket.addToBasket(3); // Add 3 mangoes to the basket

    // Print the number of fruits of each type
    cout << "Number of apples: " << appleBasket.getNumApples() << endl;
```

```
    cout << "Number of mangoes: " << mangoBasket.getNumMangoes() << endl;

    // Print the total number of fruits in the basket
    cout << "Total number of fruits in the basket: " << fruitBasket.getTotalFruits() << endl;

    return 0;
}
```

We want to calculate the total marks of each student of a class in Physics, Chemistry and Mathematics and the average marks of the class. The number of students in the class are entered by the user. Create a class named Marks with data members for roll number, name and marks. Create three other classes inheriting the Marks class, namely Physics, Chemistry and Mathematics, which are used to define marks in individual subject of each student. Roll number of each student will be generated automatically.

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

// Base class Marks
class Marks {
protected:
    int rollNumber;
    string name;
    int physicsMarks;
    int chemistryMarks;
    int mathMarks;

public:
    Marks() {
        static int counter = 1;
        rollNumber = counter++;
        name = "";
        physicsMarks = 0;
```

```
        chemistryMarks = 0;
        mathMarks = 0;
    }

    void setName(string studentName) {
        name = studentName;
    }

    void setPhysicsMarks(int marks) {
        physicsMarks = marks;
    }

    void setChemistryMarks(int marks) {
        chemistryMarks = marks;
    }

    void setMathMarks(int marks) {
        mathMarks = marks;
    }

    int getRollNumber() const {
        return rollNumber;
    }

    int getTotalMarks() const {
        return physicsMarks + chemistryMarks + mathMarks;
    }
};

// Subclass Physics
```

```
class Physics : public Marks {
public:
    void setPhysicsMarks(int marks) {
        physicsMarks = marks;
    }
};

// Subclass Chemistry
class Chemistry : public Marks {
public:
    void setChemistryMarks(int marks) {
        chemistryMarks = marks;
    }
};

// Subclass Mathematics
class Mathematics : public Marks {
public:
    void setMathMarks(int marks) {
        mathMarks = marks;
    }
};

int main() {
    int numStudents;
    cout << "Enter the number of students in the class: ";
    cin >> numStudents;

    Physics physicsMarks;
    Chemistry chemistryMarks;
```

```
    Mathematics mathMarks;

    for (int i = 0; i < numStudents; i++) {
        string name;
        int physics, chemistry, math;

        cout << "Enter name of student " << i + 1 << ": ";
        cin.ignore();
        getline(cin, name);

        cout << "Enter Physics marks: ";
        cin >> physics;
        physicsMarks.setPhysicsMarks(physics);

        cout << "Enter Chemistry marks: ";
        cin >> chemistry;
        chemistryMarks.setChemistryMarks(chemistry);

        cout << "Enter Mathematics marks: ";
        cin >> math;
        mathMarks.setMathMarks(math);

        cout << endl;
    }

    // Calculate average marks of the class
    double totalAverage = 0;
    for (int i = 0; i < numStudents; i++) {
        totalAverage += (physicsMarks.getTotalMarks() + chemistryMarks.getTotalMarks() +
mathMarks.getTotalMarks());
```

```
    }
    double classAverage = totalAverage / (numStudents * 3);

    cout << "Average marks of the class: " << classAverage << endl;

    return 0;
}
```

We want to store the information of different vehicles. Create a class named Vehicle with two data member named mileage and price. Create its two subclasses
*Car with data members to store ownership cost, warranty (by years), seating capacity and fuel type (diesel or petrol).
*Bike with data members to store the number of cylinders, number of gears, cooling type(air, liquid or oil), wheel type(alloys or spokes) and fuel tank size(in inches)
Make another two subclasses Audi and Ford of Car, each having a data member to store the model type. Next, make two subclasses Bajaj and TVS, each having a data member to store the make-type.
Now, store and print the information of an Audi and a Ford car (i.e. model type, ownership cost, warranty, seating capacity, fuel type, mileage and price.) Do the same for a Bajaj and a TVS bike.

```
#include <iostream>

using namespace std;

// Base class Vehicle

class Vehicle {

protected:

    float mileage;

    float price;

public:

    Vehicle(float m, float p) : mileage(m), price(p) {}

    void displayInfo() {

        cout << "Mileage: " << mileage << " km/l" << endl;
```

```
        cout << "Price: $" << price << endl;
    }
};

// Subclass Car
class Car : public Vehicle {
protected:
    float ownershipCost;
    int warranty;
    int seatingCapacity;
    string fuelType;

public:
    Car(float m, float p, float oc, int w, int sc, string ft)
        : Vehicle(m, p), ownershipCost(oc), warranty(w), seatingCapacity(sc), fuelType(ft) {}

    void displayInfo() {
        Vehicle::displayInfo();
        cout << "Ownership Cost: $" << ownershipCost << endl;
        cout << "Warranty: " << warranty << " years" << endl;
        cout << "Seating Capacity: " << seatingCapacity << " seats" << endl;
        cout << "Fuel Type: " << fuelType << endl;
    }
};

// Subclass Bike
class Bike : public Vehicle {
protected:
    int numCylinders;
    int numGears;
```

```
    string coolingType;
    string wheelType;
    float fuelTankSize;

public:
    Bike(float m, float p, int nc, int ng, string ct, string wt, float fts)
        : Vehicle(m, p), numCylinders(nc), numGears(ng), coolingType(ct), wheelType(wt),
fuelTankSize(fts) {}

    void displayInfo() {
        Vehicle::displayInfo();
        cout << "Number of Cylinders: " << numCylinders << endl;
        cout << "Number of Gears: " << numGears << endl;
        cout << "Cooling Type: " << coolingType << endl;
        cout << "Wheel Type: " << wheelType << endl;
        cout << "Fuel Tank Size: " << fuelTankSize << " inches" << endl;
    }
};

// Subclass Audi
class Audi : public Car {
private:
    string modelType;

public:
    Audi(float m, float p, float oc, int w, int sc, string ft, string mt)
        : Car(m, p, oc, w, sc, ft), modelType(mt) {}

    void displayInfo() {
        cout << "Audi Car Information:" << endl;
```

```
        cout << "Model Type: " << modelType << endl;
        Car::displayInfo();
    }
};

// Subclass Ford
class Ford : public Car {
private:
    string modelType;

public:
    Ford(float m, float p, float oc, int w, int sc, string ft, string mt)
        : Car(m, p, oc, w, sc, ft), modelType(mt) {}

    void displayInfo() {
        cout << "Ford Car Information:" << endl;
        cout << "Model Type: " << modelType << endl;
        Car::displayInfo();
    }
};

// Subclass Bajaj
class Bajaj : public Bike {
private:
    string makeType;

public:
    Bajaj(float m, float p, int nc, int ng, string ct, string wt, float fts, string mt)
        : Bike(m, p, nc, ng, ct, wt, fts), makeType(mt) {}
```

```
    void displayInfo() {
        cout << "Bajaj Bike Information:" << endl;
        cout << "Make Type: " << makeType << endl;
        Bike::displayInfo();
    }
};

// Subclass TVS
class TVS : public Bike {
private:
    string makeType;

public:
    TVS(float m, float p, int nc, int ng, string ct, string wt, float fts, string mt)
        : Bike(m, p, nc, ng, ct, wt, fts), makeType(mt) {}

    void displayInfo() {
        cout << "TVS Bike Information:" << endl;
        cout << "Make Type: " << makeType << endl;
        Bike::displayInfo();
    }
};

int main() {
    // Creating an Audi car object
    Audi audi(10.5, 50000, 1500, 3, 5, "Petrol", "A3");
    audi.displayInfo();

    cout << endl;
```

```
    // Creating a Ford car object
    Ford ford(9.5, 45000, 1300, 2, 4, "Diesel", "Figo");
    ford.displayInfo();

    cout << endl;

    // Creating a Bajaj bike object
    Bajaj bajaj(40, 60000, 2, 5, "Air", "Alloys", 15, "Pulsar");
    bajaj.displayInfo();

    cout << endl;

    // Creating a TVS bike object
    TVS tvs(35, 55000, 1, 4, "Liquid", "Spokes", 12, "Apache");
    tvs.displayInfo();

    return 0;
}
```

Create a class named Shape with a function that prints "This is a shape". Create another class named Polygon inheriting the Shape class with the same function that prints "Polygon is a shape". Create two other classes named Rectangle and Triangle having the same function which prints "Rectangle is a polygon" and "Triangle is a polygon" respectively. Again, make another class named Square having the same function which prints "Square is a rectangle". Now, try calling the function by the object of each of these classes.

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

// Base class Shape
class Shape {
public:
    void print() {
```

```
        cout << "This is a shape." << endl;
    }
};

// Subclass Polygon inheriting Shape
class Polygon : public Shape {
public:
    void print() {
        cout << "Polygon is a shape." << endl;
    }
};

// Subclass Rectangle inheriting Polygon
class Rectangle : public Polygon {
public:
    void print() {
        cout << "Rectangle is a polygon." << endl;
    }
};

// Subclass Triangle inheriting Polygon
class Triangle : public Polygon {
public:
    void print() {
        cout << "Triangle is a polygon." << endl;
    }
};

// Subclass Square inheriting Rectangle
class Square : public Rectangle {
```

```
public:
    void print() {
        cout << "Square is a rectangle." << endl;
    }
};

int main() {
    Shape shape;
    shape.print(); // Output: This is a shape.

    Polygon polygon;
    polygon.print(); // Output: Polygon is a shape.

    Rectangle rectangle;
    rectangle.print(); // Output: Rectangle is a polygon.

    Triangle triangle;
    triangle.print(); // Output: Triangle is a polygon.

    Square square;
    square.print(); // Output: Square is a rectangle.

    return 0;
}
```

All the banks operating in India are controlled by RBI. RBI has set a well defined guideline (e.g. minimum interest rate, minimum balance allowed, maximum withdrawal limit etc) which all banks must follow. For example, suppose RBI has set minimum interest rate applicable to a saving bank account to be 4% annually; however, banks are free to use 4% interest rate or to set any rates above it.

Write a program to implement bank functionality in the above scenario. Note: Create few classes namely Customer, Account, RBI (Base Class) and few derived classes

(SBI, ICICI, PNB etc). Assume and implement required member variables and functions in each class.

Hint:

```
Class Customer
{
//Personal Details ...
// Few functions ...
}
Class Account
{
// Account Detail ...
// Few functions ...
}
Class RBI
{
Customer c; //hasA relationship
Account a;        //hasA relationship
..
Public double GetInterestRate() {  }
Public double GetWithdrawalLimit() {       }
}
Class SBI: public RBI
{
//Use RBI functionality or define own functionality.
}
Class ICICI: public RBI
{
//Use RBI functionality or define own functionality.
}
```

```
#include <iostream>

using namespace std;


// Class Customer

class Customer {

   // Personal Details and functions related to customer...

};


// Class Account

class Account {

   // Account Details and functions related to account...

};
```

```
// Base Class RBI
class RBI {
protected:
   Customer c; // hasA relationship with Customer
   Account a;  // hasA relationship with Account

public:
   virtual double GetInterestRate() {
      // RBI's guideline for getting the interest rate
      return 4.0; // Example: Minimum interest rate set by RBI
   }

   virtual double GetWithdrawalLimit() {
      // RBI's guideline for getting the withdrawal limit
      return 50000.0; // Example: Maximum withdrawal limit set by RBI
   }
};

// Derived Class SBI
class SBI : public RBI {
   // Use RBI's functionality or define its own functionality...
};

// Derived Class ICICI
class ICICI : public RBI {
   // Use RBI's functionality or define its own functionality...
};
```

```
int main() {
    // Create objects of SBI and ICICI
    SBI sbi;
    ICICI icici;

    // Accessing RBI's functionality from SBI and ICICI objects
    double sbiInterestRate = sbi.GetInterestRate();
    double iciciInterestRate = icici.GetInterestRate();

    double sbiWithdrawalLimit = sbi.GetWithdrawalLimit();
    double iciciWithdrawalLimit = icici.GetWithdrawalLimit();

    cout << "SBI Interest Rate: " << sbiInterestRate << "%" << endl;
    cout << "ICICI Interest Rate: " << iciciInterestRate << "%" << endl;

    cout << "SBI Withdrawal Limit: Rs. " << sbiWithdrawalLimit << endl;
    cout << "ICICI Withdrawal Limit: Rs. " << iciciWithdrawalLimit << endl;

    return 0;
}
```