// Problem- 1: Single Inheritance Problem --->

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

class Shape {
protected:

	int height = 3;
	int width = 4;

public:

void setHeight(int x) {
	height = x;
}

void setWidth(int y) {
	width = y;
}

};

class Rectangle : public Shape {
public:

	int area() {
		int result = this->height * this->width;
		return result;
	}

	int perimeter() {
		int result = 2 * this->height + 2 * this->width;
		return result;
	}
};

int main() {

	Rectangle r1;
	Rectangle r2;

	cout << "Height = "<< endl;
	int p;
```

```
        cin >> p;

        cout <<"Width = " << endl;
        int q;
        cin >> q;

        r1.setHeight(p);
        r2.setWidth(q);

        int result1 = r1.area();
        cout <<"Area = " << result1 << endl;
        int result2 = r2.perimeter();
        cout <<"Perimeter = " <<result2 << endl;

}
```

// Problem- 2: Multilevel Inheritance Problem --->

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

class Vehicle {

protected:
        string fuelType = "Disel";
        int capacity = 100;

public:
        virtual void displayDetails() {
                cout << "Fuel Type: " << fuelType << endl;
                cout << "Capacity: " <<capacity << endl;
        }
};

class Car : public Vehicle {

protected:
        int numSeats = 30;
public:
        void displayDetails() override {
                Vehicle:: displayDetails();
```

```
        cout << "Number of seats: " << numSeats << endl;
    }
};

class Sedan : public Car {

private:
    int bootSpace = 500;

public:
    void displayDetails() override {
        Car:: displayDetails();
        cout << "Extra boot space for Sedan: " << bootSpace << endl;
    }
};

int main() {
    Vehicle v1;
    Car c2;
    Sedan s3;

    cout <<"Vehicle Details: " << endl;
    v1.displayDetails();
    cout << endl;

    cout <<"Car details :" << endl;
    c2.displayDetails();
    cout << endl;

    cout << "Sedan details: " << endl;
    s3.displayDetails();
    cout << endl;
}
```

// Problem- 3: Multiple Inheritance Problem --->

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

class Student {
```

```
protected:
	string name;
	int rollNumber;

public:
	void getStudent(string Name, int roll) {
		name = Name;
		rollNumber = roll;
	}
};

class Test {

protected:
	int marks;
	char grade;

public:
	void getTest(int testMarks, char Grade) {
		marks = testMarks;
		grade = Grade;
	}
};

class Result : public Student, public Test {

public:
	void details() {
		cout <<"Student's name: " << name << endl;
		cout <<"Students roll: " << rollNumber << endl;
		cout <<"Students marks: " << marks << endl;
		cout << "Student grade: " << grade << endl;
	}
};

int main() {
	Result r1;

	r1.getStudent("Rain", 22);
	r1.getTest(199, 'A');
	r1.details();
}
```

// Problem- 4: Hybrid Inheritance Problem --->

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

using namespace std;

class Account {
protected:
	string accountNumber;
	double balance;

public:
	Account(string accNumber, double initialBalance) {
		accountNumber = accNumber;
		balance = initialBalance;
	}

	virtual void deposit(double amount) {
		balance += amount;
		cout << "Deposited " << amount << " into account " << accountNumber << endl;
	}

	virtual void withdraw(double amount) {
		if (balance >= amount) {
			balance -= amount;
			cout << "Withdrawn " << amount << " from account " << accountNumber
<< endl;
		} else {
			cout << "Insufficient funds in account " << accountNumber << endl;
		}
	}

	virtual void displayBalance() const {
		cout << "Account " << accountNumber << " has a balance of " << balance <<
endl;
	}
};

class SavingsAccount : virtual public Account {
public:
```

```
    SavingsAccount(const string& accNumber, double initialBalance) : Account(accNumber,
initialBalance) {}

    void deposit(double amount) override {
        Account::deposit(amount);
    }

    void withdraw(double amount) override {
        Account::withdraw(amount);
    }
};

class CurrentAccount : virtual public Account {
public:
    CurrentAccount(const string& accNumber, double initialBalance) : Account(accNumber,
initialBalance) {}

    void deposit(double amount) override {
        Account::deposit(amount);
    }

    void withdraw(double amount) override {
        Account::withdraw(amount);
    }
};

class JointAccount : public SavingsAccount, public CurrentAccount {
public:
    JointAccount(const string& accNumber, double initialBalance)
        : Account(accNumber, initialBalance), SavingsAccount(accNumber,
initialBalance), CurrentAccount(accNumber, initialBalance) {}

    void deposit(double amount) override {
        SavingsAccount::deposit(amount);
        CurrentAccount::deposit(amount);
    }

    void withdraw(double amount) override {
        SavingsAccount::withdraw(amount);
        CurrentAccount::withdraw(amount);
    }

    void displayBalance() const override {
```

```
                cout << "Joint Account " << accountNumber << " has a savings balance of " <<
balance
                    << " and a current balance of " << balance << endl;
        }
};

int main() {
        SavingsAccount savings("SA123", 1000);
        CurrentAccount current("CA456", 2000);
        JointAccount joint("JA789", 3000);

        savings.deposit(500);
        savings.displayBalance();

        current.withdraw(100);
        current.displayBalance();

        joint.deposit(1000);
        joint.displayBalance();

        joint.withdraw(500);
        joint.displayBalance();

        return 0;
}
```

// Problem- 5: Hierarchical Inheritance Problem --->

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

using namespace std;

class Animal {
protected:
        string name;
        string sound;

public:
        Animal(const string& animalName, const string& animalSound) : name(animalName),
sound(animalSound) {}
```

```
	virtual void makeSound() const {
		cout << name << " says: " << sound << endl;
	}
};

class Dog : public Animal {
public:
	Dog(const string& dogName) : Animal(dogName, "Woof") {}
};

class Cat : public Animal {
public:
	Cat(const string& catName) : Animal(catName, "Meow") {}
};

class Cow : public Animal {
public:
	Cow(const string& cowName) : Animal(cowName, "Moo") {}
};

int main() {
	Dog dog("Buddy");
	Cat cat("Whiskers");
	Cow cow("Bessie");

	dog.makeSound();
	cat.makeSound();
	cow.makeSound();

	return 0;
}
```