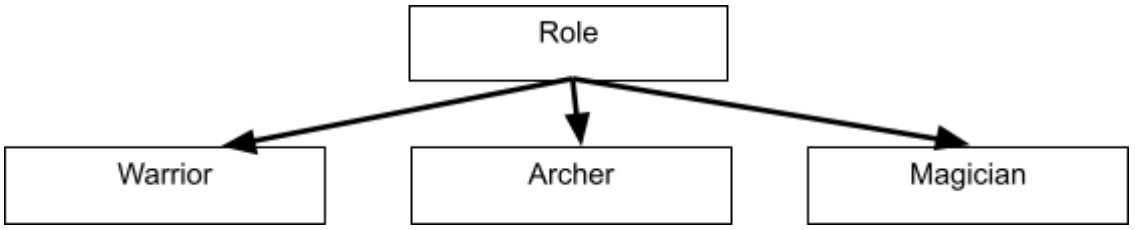
Lab 6 Inheritance and Polymorphism

For this problem, we will implement inheritance and polymorphism.

The architecture is shown as the following figure.

We have a base class Role, and three derived classes



● Class Role

- Role(string ID, int level, int money, int hp)
  Public constructor.
  - ID: Role's name.
  - level: Role's level.
  - money: Amount of money that the role has.
  - hp: Role's hp.
- void PrintAllInfo()
  Print this character's name, level, money and hp. You can see the format in the sample output.
- virtual void ShowRole()
  Pure virtual function.
- String getID()
  Public function to get the Role's ID. (name)

● Class Warrior

- Warrior (string ID, int level, int money, int hp)
  Public constructor. This class inherits the class Role.
- void ShowRole()
  Print "Role's name is Warrior "

● Class Archer

- Archer (string ID, int level, int money, int hp)
  Public constructor. This class inherits the class Role.
- void ShowRole ()
  Print "Role's name is Archer "

● **Class Magician**

- Magician(string ID, int level, int money, int hp)
  Public constructor. This class inherits the class Role.
- void ShowRole ()
  Print “Role’s name is Magician "

Class Staff

Private:
  Bool isMagician(Role* role)
    Check whether the dynamic casting is performed successfully.

Public:
  Staff(string staffName, int damage)
    staffName: The name of the staff
    damage: The damage that the staff do
  Bool equipable(Role* role);
    Check whether the role can equip this staff.

● **Function**

- bool isMagician(Role* role)
  In this function, we would like to assign a pointer of a base class type (Role) to a pointer of its derived class type (Magician). The advantage of dynamic casting is that it can be used to check whether casting is performed successfully.
  Implement dynamic casting, and return true and print ” Player is a magician” if casting success. Return False and print out “bad cast” if casting failed. Using this function to check whether the player is a magician
- bool equipable(Role* role)
  In this function, you need to use the isMagician to check whether the player is magician, if the player is magician then he/she can equip the staff.

Example :

```
Role* player1 = Warrior("Steve", 1, 1987, 200);
Staff* staff = Staff("Flame",10);
Staff->equiable(player1);
cout << endl;

player1 = Magician("Lisa", 15, 981, 145);
Staff->equiable(player2);
cout << endl;
```

Output will be :

bad cast.

Lisa is a magician.
Lisa can equip this staff.

* **Hint:**
    1. We will provide templates (main.cpp, Warrior.cpp, Warrior.h, and staff.h) .
    2. Please declare the class in the header file respectively.

Input Format

The number in the first line is the number of test cases
Then, each following line contains “ID, level, money, HP” about that character

Output Format

Print who can equip the staff(class == Magician), and their character information.

Sample Input & Output

INPUT :
4
Steve, Warrior, 1, 1987, 200
Legolas, Archer, 99, 99999, 6000
Lisa, Magician, 15, 981, 145
Rose, Magician, 30, 234, 668

OUTPUT:
ID is : Steve
Level is : 1
Money is : 1987
Hp is : 200
Role Steve is Warrior

ID is : Legolas
Level is : 99
Money is : 99999
Hp is : 6000
Role Legolas is Warrior

ID is : Lisa
Level is : 15
Money is : 981
Hp is : 145

Role Lisa is Magician

ID is : Rose
Level is : 30
Money is : 234
Hp is : 668
Role Rose is Magician

Checking whether the role can equip a staff..
error bad_cast
error bad_cast
Lisa can equip staff_flame.
Rose can equip staff_flame.