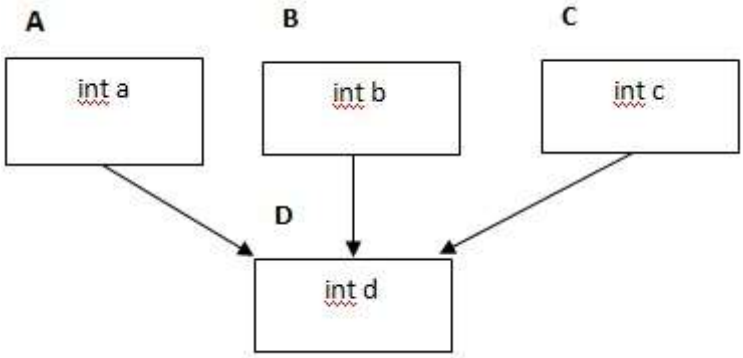
تمرین ارث بری

با ارث بری ساده و چند سطحی آشنایی دارید. حال در این سوال میخواهیم چند مدل دیگر از ارث بری را پیاده سازی و تمرین کنیم. در هر قسمت میبایست با توجه به شماتیک داده شده و توضیحات ، کلاس های مربوطه را طراحی کنید:

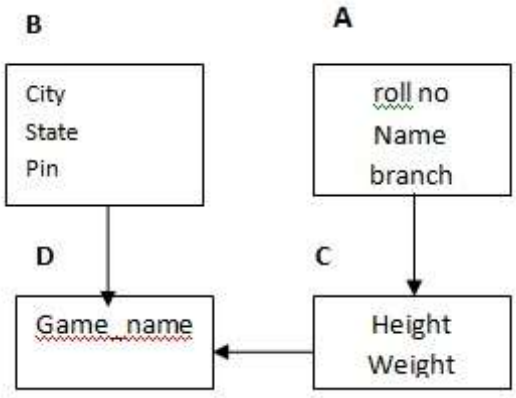
۱. **Multiple Inheritances**

هنگامی که یک کلاس از بیش از یک کلاس base ارث میبرد



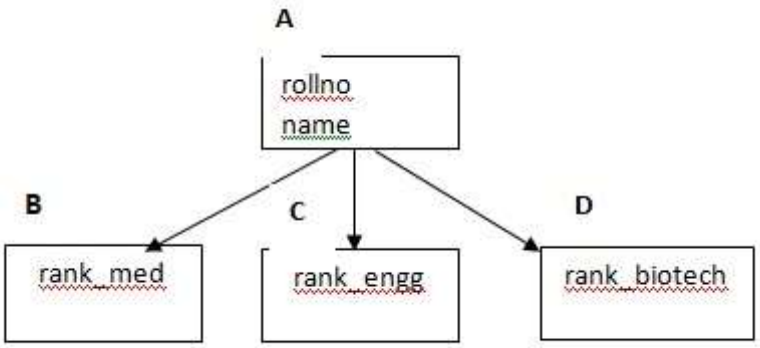
۲. **Hybrid Inheritance**

ترکیب بیش از یک نوع از ارث بری



۳. **Hierarchical Inheritance**

وقتی بیش از یک کلاس از یک کلاس base ارث میبرند


A
rollno
name
B
C
D
rank_med
rank_engg
rank_biotech

چند مضربی

به شما سه کلاس A , B , C داده میشود . هر سه این کلاس ها تابع func مخصوص به خود را دارند.

در کلاس A : این تابع مقداری که به آن پاس داده میشود را دوبرابر میکند.

```
class A
{
    public:
        A(){
            callA = 0;
        }
    private:
        int callA;
        void inc(){
            callA++;
        }

    protected:
        void func(int & a)
        {
            a = a * 2;
            inc();
        }
    public:
        int getA(){
            return callA;
        }
};
```

در کلاس B : این تابع مقداری که به آن پاس داده میشود را سه برابر میکند.

```
class B
{
    public:
        B(){
            callB = 0;
```

```
        }
    private:
        int callB;
        void inc(){
            callB++;
        }
    protected:
        void func(int & a)
        {
            a = a * 3;
            inc();
        }
    public:
        int getB(){
            return callB;
        }
};
```

در کلاس C : این تابع مقداری که به آن پاس داده میشود را پنج برابر میکند.

```
class C
{
    public:
        C(){
            callC = 0;
        }
    private:
        int callC;
        void inc(){
            callC++;
        }
    protected:
        void func(int & a)
        {
            a = a * 5;
            inc();
        }
    public:
        int getC(){
            return callC;
```

```
        }
};
```

به شما کلاس D به فرم زیر داده میشود :

```
class D
{

    int val;
    public:
        //Initially val is 1
         D()
         {
             val = 1;
         }

         //Implement this function
         void update_val(int new_val)
         {

         }
         //For Checking Purpose
         void check(int); //Do not delete this line.
};
```

حال شما باید تابع update_val را بگونه ای بنویسید که مقدار val موجود در کلاس D را به new_val تغییر دهد . این کار باید تنها با صدا زدن توابع $*func*$ موجود در کلاس های A , B , C انجام دهید. تضمین میشود که new_val تنها مضربی از 2 ، 3 ، 5 باشد .

ورودی :

حاوی تنها یک خط است که مقدار new_val را از کاربر میگیرد

خروجی :

به صورت اتوماتیک توسط کد زیر خروجی داده خواهد شد که فرمت نمونه آن در مثال ها موجود میباشد .

مثال :

ورودی :

```
30
```

خروجی :

```
Value = 30
A's func called 1 times
B's func called 1 times
C's func called 1 times
```

توضیحات :

در ابتدا مقدار val یک میباشد . سپس تابع func موجود در کلاس A اجرا میشود و و مقدار val دوبرابر میشود. سپس تابع func موجود در کلاس B اجرا میشود و مقدار val سه برابر میشود. سپس تابع func موجود در C اجرا میشود و این بار 5 برابر میشود که میشود 30 !

کد خروجی شما باید به فرم زیر باشد :

```
#include<iostream>

using namespace std;

class A
{
    public:
        A(){
            callA = 0;
        }
    private:
        int callA;
        void inc(){
            callA++;
```

```
        }

    protected:
        void func(int & a)
        {
            a = a * 2;
            inc();
        }
    public:
        int getA(){
            return callA;
        }
};

class B
{
    public:
        B(){
            callB = 0;
        }
    private:
        int callB;
        void inc(){
            callB++;
        }
    protected:
        void func(int & a)
        {
            a = a * 3;
            inc();
        }
    public:
        int getB(){
            return callB;
        }
};

class C
{
    public:
        C(){
```

```
            callC = 0;
        }
    private:
        int callC;
        void inc(){
            callC++;
        }
    protected:
        void func(int & a)
        {
            a = a * 5;
            inc();
        }
    public:
        int getC(){
            return callC;
        }
};

/*****************************/

class D
{

    int val;
    public:
        //Initially val is 1
         D()
         {
             val = 1;
         }

         //Implement this function
         void update_val(int new_val)
         {

         }
         //For Checking Purpose
         void check(int); //Do not delete this line.
};
```

```
/*****************************/

void D::check(int new_val)
{
    update_val(new_val);
    cout << "Value = " << val << endl << "A's func called " << getA() << " tim
}

int main()
{
    D d;
    int new_val;
    cin >> new_val;
    d.check(new_val);

}
```

Pay Check

هدفمان از این برنامه چاپ کردن چک های پرداخت یک شرکت می باشد. برای به انجام رساندن این وظیفه شما نیاز به ساختن و مشتق ساختن چندین کلاس خواهید بود. ابتدا در فایل های Employee.h و Employee.cpp یک کلاس base به نام Employee تعریف کنید که دارای اطلاعات زیر می باشد:

- نام کارمند
- شماره تامین اجتماعی (Social Security Number) به فرمت xxx-xx-xxx که x می تواند بین 0 تا 9 باشد.
- شماره ی کارمندی به فرمت xxx-L که x عددی بین 0 تا 9 و L حرفی بین A تا M می باشد.
- توابع constructor و member function های مناسبی برای اداره و دستکاری کردن data member ها

** اعتبار سنجی ورودی: تنها شماره تامین اجتماعی و شماره ی کارمندی معتبر با توجه به فرمت های بالا را دریافت کنید.

** توجه کنید این اعتبار سنجی باید در توابع setter شما باشد.

سپس در فایل های EmployeePay.h و EmployeePay.cpp کلاسی به نام EmployeePay تعریف نمایید که از کلاس Employee مشتق می شود. این کلاس باید دارای اطلاعات زیر باشد:

- پرداخت سالانه
- پرداخت هفتگی (که از روی پرداخت سالانه حساب میشود)
- توابع Constructor و member function های مناسبی برای اداره و دستکاری کردن data member ها

** اعتبار سنجی ورودی: مقادیر منفی را نباید برای پرداخت سالانه دریافت کنید.

حال در فایل های HourlyPay.h و HourlyPay.cpp کلاسی به نام HourlyPay تعریف نمایید. این کلاس باید از کلاس EmployeePay ارث ببرد و اطلاعات زیر را در خود ذخیره کند.

- نرخ پرداخت ساعتی

- نرخ پرداخت اضافه کاری(هفته ی کاری را ۴۰ ساعت در نظر بگیرید) (در صورتی که ساعت کار هفتگی بیشتر از 40 ساعت باشد مقدار آن ۱.۵ برابر نرخ پرداخت ساعتی خواهد بود)
- تعداد ساعت های کار کرده
- توابع Constructor و member function های مناسبی برای اداره و دستکاری کردن data member ها

** اعتبار سنجی ورودی: مقادیر بیشتر از 60 را برای تعداد ساعات کارکرده قبول نکنید.

در آخر هم باید در فایل main با استفاده از کلاس هایی که نوشتید برنامه ای بنویسید که با استفاده از تابعی به نام PrintCheck(HourlyPay) چک پرداختی را چاپ نماید. این برنامه باید با گرفتن داده از ورودی اطلاعات کارمند را پر کرده و سپس حقوق کارمند را محاسبه کرده و چک دستمزد او را نمایش دهد.