توابع مجازی

در این سوال قصد داریم تا شمارا با توابع مجازی آشنا کنیم . سه کلاس *Professor* , *Person* و *Student* ایجاد کنید.

در کلاس *Person*_دو عضو داده ای سن (*age_*) و نام (*name*) وجود دارد. دو کلاس *Professor* و *Student* از کلاس *Person* ارث میبرند.

در کلاس *Professor* باید دو عضو داده ای *publications* و curid وجود داشته باشد همچنین دو تابع عضو زیر وجود دارند :

- تابع *getdata* : وظیفه ی گرفتن اطلاعات استاد شامل نام و سن و انتشارات استاد میباشد.
- تابع *putdata* : وظیفه ی نمایش اسم ، سن ، انتشارات و curid را دارد.

در کلاس *Student* دو عضو داده ای نمره (*marks*) که یک آرایه 6 تایی و curid داریم. در این کلاس دو تابع عضو زیر وجود دارند :

- تابع *getdata* : وظیفه ی گرفتن اطلاعات دانشجو شامل نام و سن و نمرات دانشجو در شش درس میباشد.
- تابع *putdata* : وظیفه ی نمایش اسم ، سن ، مجموع نمرات ، curid را دارد.

برای هر شئی که از کلاس استاد یا دانشجو ساخته میشود curid یک عدد ترتیب افزایشی که از یک شروع میشود به آن ها داده شوند.

این مساله را با توابع مجازی ، سازنده و مقادیر static حل کنید. شما میتوانید عضو های داده ای بیشتری در کلاس ها بنابر نیاز ایجاد کنید.

ورودی :

خط اول ورودی تعداد شی هایی است که میسازیم سپس درصورتی که کاربر در خط بعدی عدد یک را وارد کرد به معنای ساخت شئی استاد و درصورتی که دو را وارد کرد به معنای ساخت شئی دانشجو میباشد.

بعد از وارد کردن نوع شئی در خط بعدی موارد مورد نیاز هر شئی از کاربر گرفته میشود . برای مثال برای استاد به ترتیب اسم ، سن ، انتشارات گرفته میشود یا درصورتی که دانشجو بود به ترتیب اسم ، سن ، نمرات 6 درس گرفته میشود.

خروجی :

دو نوع خروجی بر اساس نوع شئی ها داریم :

۱. استاد : مشخصات آن استاد را با یک اسپیس در خروجی چاپ کند.
۲. دانشجو : مشخصات آن دانشجو را با یک اسپیس در خروجی چاپ کند.

مثال :

ورودی :

```
4
1
Walter 56 99
2
Jesse 18 50 48 97 76 34 98
2
Pinkman 22 10 12 0 18 45 50
1
White 58 87
```

خروجی :

```
Walter 56 99 1
Jesse 18 403 1
Pinkman 22 135 2
White 58 87 2
```

کد main :

```
int main(){

    int n, val;
    cin>>n; //The number of objects that is going to be created.
    Person *per[n];

    for(int i = 0;i < n;i++){

        cin>>val;
        if(val == 1){
            // If val is 1 current object is of type Professor
            per[i] = new Professor;

        }
        else per[i] = new Student; // Else the current object is of type Stude

        per[i]->getdata(); // Get the data from the user.

    }

    for(int i=0;i<n;i++)
        per[i]->putdata(); // Print the required output for each object.

    return 0;

}
```

Abstract Cache

کلاس های انتزاعی در C++ تنها میتوانند به عنوان کلاس های پایه باشند پس در نتیجه آن ها میتوانند توابعی مجازیی داشته باشند که تعریف نشده باشند.

کش (cache) یک کامپوننتی است که در آن اطلاعات ذخیره میشود به صورتی که درخواستی های در آینده به این اطلاعات سریعتر پاسخ داده شود !

اطلاعات موجود در کش ممکن است ناشی از محاسبات پیشین یا تکراری از اطلاعات ذخیره شده در جای دیگر باشد.

هنگامی که داده های درخواست شده در کش موجود باشد عمل cache hit رخ داده است . و وقتی که موجود نباشد عمل cache miss به آن میگویند.

در عمل cache hit خواندن داده بسیار سریعتر نسبت به محاسبات مجدد یا خواندن اطلاعات از دیسک یا .... میباشد.

پس در نتیجه اگر ما درخواست های بیشتری را بتوانیم توسط cache پاسخگو باشیم سرعت بیشتری به سیستم خود میبخشیم .

سیاست های مختلفی برای قرار دادن درخواست ها در صف موجود در کش وجود دارد . یکی از این سیاست ها "LEAST RECENTLY USED" یا به صورت خلاصه "LRU" میباشد و به این صورت میباشد که همیشه جدیدترین درخواست در اول صف قرار میگیرد و درخواست های قدیمی تر به عقب برگردانده میشوند .

برای مثال ، اگر ما یک کش با توانایی ذخیره 5 کلید داشته باشد و حالت اولیه کش ما به فرم زیر باشد :

```
5 3 2 1 4
```

حال اگر کلید بعدی که درخواست شود "1" باشد صف موجود در کش ما به صورت زیر میشود :

```
1 5 3 2 4
```

و اگر کلید بعدی که درخواست میشود "6" باشد صف موجود در کش ما به صورت زیر میشود :

```
6 1 5 3 2
```

قابل مشاهده است که کلید 4 از آخر صف ما به دلیل اینکه حجم حافظه کش ما 5 بوده است و توانایی نگهداری آن دیگر ممکن نبوده حذف شده است.

به شما کلاس انتزاعی *cache* با عضو های داده ای و توابع عضو زیر داده میشود :

- cp : حجم حافظه ی کش
- set() : اضافه کردن کلید جدید به صف کش
- get() : مقدار موجود برای آن کلید در کش درصورتی که موجود باشد. درصورتی که آن کلید در کش موجود نباشد مقدار منفی یک بر میگرداند.

با توجه به سوال گفته شده شما نیازمند یک ساختمان داده ای هستید که هر کلید را به یک داده نظیر کند ! انتخاب و یا ساخت این ساختمان داده به اختیار شماست .

کد کلاس Cache :

```
class Cache{

   protected:
   //data structure
   int cp;  //capacity
   virtual void set(int, int) = 0; //set function
   virtual int get(int) = 0; //get function

};
```

ورودی :

در خط اول عدد N و M به شما داده میشود که به ترتیب نمایانگر تعداد دستورات Set/Get و حجم حافظه کش میباشد. در N خط بعدی ابتدا اسم دستور و سپس مواردی که برای هر دستور نیاز است وارد میشود.

در ابتدا کش کاملا خالیست و هیچ داده ای در آن وجود ندارد.

خروجی :

خروجی کد شما درواقع خروجی حاصل از دستور Get میباشد.

مثال :

ورودی :

```
3 1
set 1 2
get 1
get 2
```

خروجی :

```
2
-1
```

اَشکال یا اِشکال

برنامه ای بنویسید برای اشکال مختلف . اشکال زیر در این برنامه وجود دارند :

۱. مثلث
۲. مستطیل
۳. لوزی
۴. مربع
۵. 6ضلعی
۶. 8 ضلعی

هر یک از این اشکال با مجموعه ای از نقاط بر روی مختصات دکارتی مشخص میشود.پس ما در هر کلاس مجموعه ای از کلاسِ نقاط(اعشاری) داریم ! از طرفی دیگر هر شکل دارای یک رنگ میباشد که در هنگام ساخت (سازنده) هر شی رنگ آن را مشخص میکنیم. اسم هر رنگ نهایتا 16 کاراکتر است. حال با کمک این نقاط وظیفه ی ما محاسبه موارد زیر برای هر شکل میباشد :

۱. محیط
۲. مساحت
۳. طول قطرها برای اشکال بیش تر از 4 ضلعی مرتب شده نزولی

در این برنامه ما ابتدا شکلی که قصد داریم بسازیم انتخاب میکنیم و سپس نقاط مربوط به شکل هارا از کاربر گرفته و تمام موارد بالا را در خروجی برای آن شکل نشان میدهیم.

ورودی

در خط اول تعداد شکل ها می آید . سپس در خط بعدی نوع تمام شکل هارا دریافت میکنیم . و بعد از آن برای هر شکل نقاط لازمه را دریافت میکنیم !

خروجی

برای هر شکل باید سه مورد ذکر شده تا *دو رقم اعشار* به فرم مثال ها نشان داده شود.

نکات :

۱. تمامیه متغییر ها باید عضو خصوصی باشند.

۲. کلاس های شما باید به بهترین نوع ممکن ارث برده باشند.

مثال

ورودی:

```
3
1 2 3
GREEN : 0:0 2:0 1:4
RED : 0:0 0:8 2:8 2:0
LIGHTBLUE : 0:2 1:0 2:2 1:4
```

خروجی:

```
GREEN : 10.25 4.00
RED : 20.00 16.00 8.25 8.25
LIGHTBLUE : 8.94 4.00 4.00 2.00
```