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
```