پوینتر بی‌مصرف

برنامه‌ای بنویسید که در ابتدا دو عدد دریافت کند، تعدادِ سطرها و تعدادِ ستون‌ها. سپس یک آرایه با مشخّصاتِ گفته شده از کاربر دریافت کند. خروجیِ این برنامه، یک آرایه‌ی جدید است، که عنصری از آن که در سطرِ iام و ستونِ jام قرار دارد برابر است با مجموعِ اعدادِ سطرِ iام و ستونِ jام در آرایه‌ی قبل.

توجه کنید که آرایه‌تان را با پوینتر بسازید و از پوینتر استفاده کنید! در غیر این صورت نمره‌ای به شما تعلق نخواهد گرفت!

مثال

ورودی نمونه

```
3 4
1 2 3 4
5 6 7 8
9 10 11 12
```

خروجی نمونه

```
24 26 28 30
36 38 40 42
48 50 52 54
```

ضرب ماتریس

برنامه‌ای بنویسید که از کاربر دو ماتریس گرفته و ضرب آن را حساب کند.

ورودی

در خط اول ورودی ابعاد اولین ماتریس و سپس خود ماتریس و به همین شکل ماتریس دوم وارد می‌شود!

خروجی

حاصل‌ضرب دو ماتریس را چاپ شود. در صورتیکه ضرب امکان‌پذیر نبود `error` چاپ شود.

توجه کنید که آرایه‌تان را با پوینتر بسازید و از پوینتر استفاده کنید! در غیر این صورت نمره‌ای به شما تعلق نخواهد گرفت!

مثال

ورودی نمونه

```
2 3
0 1 2
3 4 5
3 2
```

```
6 7
8 9
10 11
```

خروجی نمونه

```
28 31
100 112
```

خروجی؟

فرض کنید در برنامه زیر ، متغیر c آدرس 6940 ، متغیر d آدرس 9772 و متغیر e آدرس 2224 را دارد. خروجی کد زیر چیست؟ توضیح دهید.

```
int* p1;

int* p2;

int* p3;

int  c = 100 , d = 200 , e = 300;

p3 = &e;
p2 = &d;
p1 = &c;

cout << "*p3 = " << *p3 << endl;

p3 = p1;

cout << "*p3 = " << *p3

     << ", p3 = " << p3 << endl;

*p1 = *p2;
```

```
cout << "*p1 = " << *p1

     << ", p1 = " << p1 << endl;
```

عبارات زیر را در نظر بگیرید:

```
int *p;
int i;
int k;
i = 42;
k = i;
p = &i;
```

بعد از دستورات بالا ، کدام دستور مقدار i را به 75 تغییر می‌دهد؟چرا؟

A) k = 75;
B) *k = 75;
C) p = 75;
D) *p = 75;
E) بیش از دو مورد از موارد بالا

بازم خروجی؟

خروجی کد زیر چیست ؟ توضیح دهید.

```
int count = 24, *temp , sum = 33;
temp = &count;
*temp = 8;
temp = &sum;
printf("count = %d, *temp = %d , sum = %d\n", count , *temp , sum );
```

بابا بسه دیگه!

خروجی برنامه زیر را با ذکر دلیل بنویسید.

```
int a[] = {12 , 10};
int& ar = a[0];
int* ap = a;
cout << (*a)++/ar << endl;
cout << ((*ap)^(*a+1)) << endl;
cout << *(&*(a+1)) << endl;
```

آرایه متحرک

برنامه‌ای با یک آرایه‌ی داینامیک از اعداد صحیح در نظر بگیرید که هر دفعه با گرفتن سایز یک آرایه و اعضای آن، آن را با استفاده از تابع mySort به صورت صعودی چاپ کند. برنامه تا زمانی ورودی خواهد گرفت که تعداد اعداد ۱- وارد شود. در ابتدا سایز آرایه صفر است. اگر سایز آرایه افزایش یابد باید تعدادی خانه‌ی جدید به آرایه اضافه کنید و اگر سایز کاهش یابد باید خانه‌های اضافه را پاک کنید. (تابع سورت به انتخاب شماست!) سایز آرایه بین ۱ تا ۱۰۰۰۰ می‌باشد.

صف حلقوی

یکی از ساختمان‌داده‌هایی که در پیاده‌سازی الگوریتم‌های برنامه‌نویسی بسیار کاربرد دارد، صف است. **FIFO (First in First out)** سیاست کار صف است. به این معنا که هر کسی که اول وارد صف می‌شود، اول هم خارج می‌شود. به طور کلی یک صف دارای توابع **enqueue** و **dequeue** به شرح زیر است:

- void enqueue(char element) : add an element to the end of the queue.
- char dequeue() : remove and return an element from the beginning of the queue.

صف حلقوی یک صف دایره‌ای شکل است که از **2** اشاره‌گر rear و front برای مدیریت صف استفاده می‌کند. به این صورت که rear به آخر صف و front به خانه‌ای از آرایه که نشان‌دهنده‌ی ابتدای صف است اشاره می‌کند.

در ابتدا مقدار front و rear یکی است و هر بار که یک عنصر اضافه می‌شود front یکی جلو می‌رود و هر بار که یک عنصر حذف می‌شود rear یکی جلو می‌رود. هنگامی صف پر است که مقدار front = rear -1 باشد.

توجه داشته باشید که صف شما دوار است یعنی ممکن است به ازای n عنصر فضا ، n+1 درج و n-2 حذف داشته باشیم. در این حالت اشاره‌گر front در خانه‌ی **1** و اشاره‌گر rear در خانه‌ی n-1 قرار می‌گیرد. برنامه‌ای بنویسید که یک صف حلقوی با طول **۵** را پیاده‌سازی کند و تا زمانی که کاربر کاراکتر **q** را وارد نکرده دستورهای وارد شده در ورودی را اجرا کند.

صف شما باید علاوه بر توابع بالا، توابع **rear_pos** و **front_pos** و **print** را نیز پیاده‌سازی کند.

- void print() : prints all of the current queue elements.

- int front_pos() : returns the corresponding index of the front variable.
- int rear_pos() : returns the corresponding index of the rear variable.

در این سوال برای سادگی کار هر دستور با حرف اول تابع آن شناخته می‌شود یعنی:

- print() -> p
- enqueue () -> e
- dequeue() -> d
- front_pos() -> f
- rear_pos() -> r
- quit() -> q

ورودی:

```
e a
e b
e c
e d
f
r
p
d
d
```

```
f
r
p
q
```

خروجی:

```
4
0
a b c d
a
b
4
2
c d
```