به نام خدا

درس برنامه نویسی پیشرفته

تمرین دوم

دانشکده مهندسی کامپیوتر

دانشگاه علم و صنعت ایران

استاد مرضیه ملکی مجد

نیم سال دوم ۱۴۰۱-۱۴۰۰

مهلت ارسال :

۱۴۰۰/۱۲/۱۳

مبحث:

آرایه و متد - مقدمات شی گرایی

مسئول تمارین :

آریا شهسوار

فهرست

آداب نامه تمرینات

- پاسخ تمامی سوالات تنها به زبان C# قابل قبول می باشد
- علیرغم اعتماد کامل تیم تی ای به شما دانشجویان عزیز ، تمامی کد های شما با سایر دانشجویان بصورت خودکار و توسط برنامه مقایسه خواهند شد . همچنین در طول ترم ، از تمامی پاسخ های شما ارائه گرفته خواهد شد و نحوه کار تمامی بخش های هر سوال از شما پرسیده خواهد شد ، لذا از کپی نمودن کد دوستانتان خودداری کنید و تمامی پاسخ ها ، کد خودتان باشد . همچنین از آنجایی که مشورت و هم فکری با سایر دوستان بسیار کار پسندیده و مفیدی است - برخلاف کپی کردن کد (: - در صورت هم فکری با دانشجوی (دانشجویان) ، نام وی را بصورت کامنت شده در ابتدای کد خود بنویسید .
- برای ارسال تمارین در طول ترم ، در مجموع ۷ روز می توانید تاخیر داشته باشید و در صورتی که جمع تاخیر دانشجویی بیشتر از ۷ روز شود ، تمرین وی قابل قبول نخواهد بود لذا تلاش کنید تمرینات را در زمان مقرر در سامانه آپلود کنید
- برای هر تمرین در زمان ددلاین دانشجو باید درخواست خود را مبنی بر ارسال با تاخیر به تی ای ها اعلام نماید
- در تمامی تمرینات سعی شده است که سوالات ساده تر در ابتدا و سوالات دشوار تر در انتهای فایل قرار گیرند (از ساده به دشوار مرتب شده اند)
- در صورت وجود هرگونه سوال در مورد تمرینات ، سعی کنید تا جایی که امکان دارد سوال خود را در گروه بپرسید چرا که شاید سوال شما ، سوال دوستتان نیز باشد و دوستانتان نیز بتوانند از پاسخ سوال شما بهره ببرند .

نکات تمرین سری دوم

- سوالات را در سامانه کوئرا و در قسمت تمرین سری دوم آپلود نمایید .
- در این تمرین و به طور کلی در تمرینات از این پس ، نیاز است برای تمامی توابع خود Unit Test مربوط به آن را نیز پیاده سازی کنید (برای هر تابع دو نمونه کافی است) به همین دلیل برای آپلود تمارین ، فولدر اصلی (فولدر شامل Unit Test و sln.*) را بصورت زیپ شده در سایت آپلود نمایید .
- دقت شود سوالات از این پس تست کیس نداشته و تشخیص درستی و نمره پاسخ شما با بررسی تست کیس های متفاوت و توضیح راه حلتان در زمان ارائه داده خواهد شد
- با توجه به مبحث تمرین سری دوم استفاده از هرگونه داده ساختاری بجز آرایه مانند List , Tuple , ... مجاز نیست
- تمرین شماره ۵ و۷ مربوط به مبحث مقدمات شی گرایی است و سایر سوالات مبحث آرایه و متد را پوشش داده است .
- در تمرین شماره ۵ از شما خواسته شده است منویی برای برنامه خود طراحی کنید :
  این منو باید این ویژگی را داشته باشد که تا زمانی که کاربر دستور exit را وارد نکرده است گزینه های منو به او نمایش داده شوند و با انتخاب شدن هر کدام از آن گزینه ها و وارد کردن دستور مربوط به آن عملیات مربوطه انجام شود .
- از آنجایی که هر سوال توسط یک تی ای طرح شده است ، تنها تی ای طراح آن سوال می تواند شما را بصورت دقیق راهنمایی کند به همین منظور طراح هر سوال در زیر نوشته شده است تا در صورت ابهام و پرسش در مورد هر سوال ، در صورتی که نیاز به پرسش سوال بصورت انفرادی در پیوی هست ، به تی ای مربوطه مراجعه بفرمایید
  - سوال ۱ . خانم شاهرخیان
  - سوال ۲ . خانم رضی
  - سوال ۳ . آقای شهسوار
  - سوال ۴ . خانم شاهرخیان
  - سوال ۵ . آقای فخاری
  - سوال ۶ . آقای مرادیان
  - سوال ۷ . آقای شهسوار

تمرین ۱ . آرایه خوب

به آرایه ای خوب گفته میشود که بتوان زیر مجموعه ای از این اعداد را پیدا کنید که اگر هر عنصر را در عددی صحیح ضرب کنید و همه این اعداد را باهم جمع کنید ، عدد بدست آمده ۱ شود .

ورودی :

در خط اول n که تعداد عضوهای آرایه است داده می شود .

$$0 \le n \le 10^5$$

در خط بعدی n عدد صحیح مثبت که با فاصله از هم جدا شده اند داده می شود . تضمین می شود که اعضای آرایه از نوع داده ای int هستند

خروجی :

اگر آرایه خوب بود true وگرنه false را چاپ کند.

Example 1 :

Input :

```
4
12 5 7 23
```

Output :

```
true
```

دلیل : با انتخاب زیر مجموعه {5,7} داریم :

```
5*3 + 7*(-2) = 1
```

Example 2 :

Input :

```
3
29 6 10
```

Output :

```
true
```

دلیل : با انتخاب زیر مجموعه {29,6,10} داریم :

```
29*1 + 6*(-3) + 10*(-1) = 1
```

Example 3 :

Input :

```
2
3 6
```

Output :

```
false
```

تمرین ۲ . تبدیل کننده

تابعی بازگشتی پیاده سازی کنید که بعنوان ورودی یک عدد صحیح را دریافت کرده و آن را با شرایط زبر به کلمات انگلیسی تبدیل کند :

۱. اگر عدد دریافتی بزرگتر از ۹۹۹ باشد تابع کلمه "Too Large" را برگرداند

۲ . اگر عدد دریافتی کوچکتر از ۹۹۹ باشد تابع کلمه "Too Small" را برگرداند

۳ . اگر عدد منفی باشد قبل از آن کلمه minus را اضافه کند

برای جلوگیری از هدر رفت وقت میتوانید از دو آرایه پیاده سازی شده زیر استفاده کنید

```
String[] units = { "zero", "one", "two", "three",
    "four", "five", "six", "seven", "eight", "nine",
"ten", "eleven",
    "twelve", "thirteen", "fourteen", "fifteen",
"sixteen",
    "seventeen", "eighteen", "nineteen" };
String[] tens = { "", "", "twenty", "thirty", "forty",
    "fifty", "sixty", "seventy", "eighty", "ninety" };
```

همچنین برای بخش صدگان ، از فرمت :

```
[ units - hundred ]
```

بعنوان مثال :

```
two - hundred : 200
```

Example 1 :

Input : -420

Output : minus four-hundred and twenty

Example 2 :

```
Input : 999
Output : nine-hundred and ninety nine
```

Example 3 :

```
Input : 1020
Output : Too large
```

Example 4 :

```
Input : -9945
Output : Too small
```

تمرین ۳ . پیمایش حلزونی

تابعی بنویسید که بعنوان ورودی یک ماتریس (آرایه) دو بعدی به همراه ابعادش را دریافت کرده و آن را بصورت حلزونی پیمایش کرده و مسیر پیمایش را بصورت یک string (که بین اعداد "," وجود دارد) برگرداند .

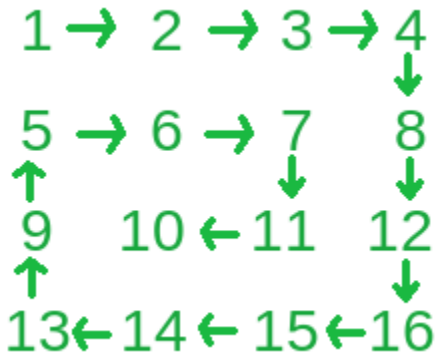
برای درک روش پیمایش حلزونی به مثال زیر توجه کنید :

Example 1 :

```
Input : row = 4 , col = 4 , Matrix[,] =
                                {{1,2,3,4},
                                {5,6,7,8},
                                {9,10,11,12},
                                {13,14,15,16}}

Output : 1,2,3,4,8,12,16,15,14,13,9,5,6,7,11,10
```

دلیل :



تمرین ۴ . مسیر خاص

علی می خواهد در مسابقات رباتیک شرکت کند و مشغول تست ربات خود میباشد. مسیر مسابقه به صورت یک مستطیل شبکه بندی شده است که نقطه ی شروع و پایان و همینطور موانع در آن مشخص است. علی می خواهد بداند که چند راه برای رسیدن از نقطه ی شروع به پایان وجود دارد به صورتی که رباتش دقیقا یک بار از روی تمامی نقاطی که در آن مانعی وجود ندارد، عبور کند.

ورودی :

در خط اول n و m داده میشود که به ترتیب تعداد ردیف ها و ستون ها است.

$$0 \le n,m \le 20$$

در هر n خط بعدی m عدد داریم که میتوانند به صورت زیر باشند:

- 1 : نشان دهنده مربع شروع است. دقیقاً یک مربع شروع وجود دارد.
- 2 : نشان دهنده مربع انتهایی است. دقیقاً یک مربع انتهایی وجود دارد.
- 0 : نشان دهنده مربع های خالی است که می توانیم روی آنها راه برویم.
- 1- : نشان دهنده موانعی است که نمی توانیم از آنها عبور کنیم.

خروجی :

تعداد مسیر های موجود.

Example 1 :

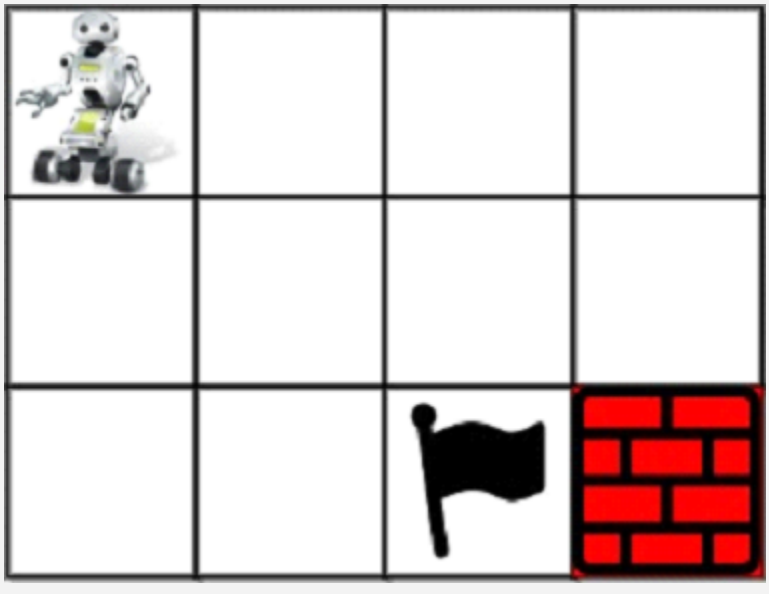
Input :

```
4 3
1 0 0 0
0 0 0 0
0 0 2 -1
```

Output:

```
2
```

دلیل :



ربات دو مسیر زیر را میتواند طی کند :

1.

```
(0,0),(0,1),(0,2),(0,3),(1,3),(1,2),(1,1),(1,0),(2,0),
(2,1),(2,2)
```

2.

```
(0,0),(1,0),(2,0),(2,1),(1,1),(0,1),(0,2),(0,3),(1,3),
(1,2),(2,2)
```

Example 2 :

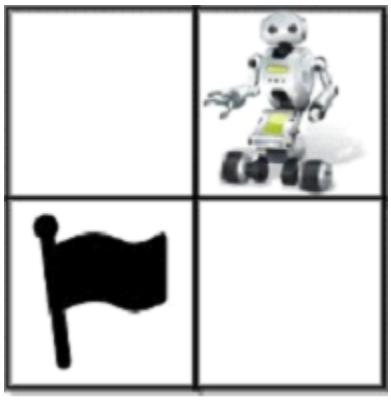
Input :

```
2 2
0 1
2 0
```

Output:

```
0
```

دلیل :



مسیری که تمام نقاط مورد نظر سوال را پوشش دهد وجود ندارد .

توجه : دقت شود که نقاط شروع و پایان می تواند در هر جایی از مستطیل باشند

تمرین ۵ . سودوکو

همانطور که میدانید جدول سودوکو از یک جدول ۹ * ۹ تشکیل شده است. در این سوال می خواهیم برنامه بنویسید که جدول سودوکو زیر را با توجه به قوانین بازی حل کند.(توضیحات بیشتر در ادامه آورده شده است)

	8	4					1	3
2				3		6		
6			5		9			2
		2				4	6	9
7								
			2	8				
	2		7					
		8			5	9		6
5				2		3		7

در این سوال شما باید کلاسی طراحی کنید که شامل فیلد های سطر، ستون و مقدار باشد که این مقدار عددی از ۱ تا ۹ میباشد. این کلاس دارای متد های Hint ، Delete ، Add و ShowTable است که در ادامه بیشتر آنها را توضیح میدهیم.

توجه ۱ : این سوال را با استفاده از شی ساخته شده از این کلاس باید حل کنید.

منوی برنامه:

- افزودن عدد در جدول (Add number)
- پاک کردن عدد (Delete)
- راهنمایی (Hint)
- نمایش جدول (Show Table)
- خروج (Exit)

روند اجرای برنامه :

۱ . افزودن عدد :

با اجرای این دستور سه عدد در یک خط از ورودی گرفته میشود که به ترتیب شماره سطر، شماره ستون و عددی که قصد داریم آن را به جدول اضافه کنیم. به عنوان نمونه میتوانید به صورت زیر عمل کنید:

Add number

[row] [column] [number]

نکته ۱ : در این قسمت تنها میتوانید به خانه هایی که در صورت سوال مقداردهی نشده اند عدد اضافه کنید و به خانه هایی که از ابتدا مقداری در خود دارند نمیتوانید عددی اضافه کنید.

نکته ۲ : برای تمیز قرار دادن میان مقادیر پیش فرض جدول و مقادیر ورودی کاربر، ورودی های کاربر را با رنگ دیگری در جدول ثبت کنید .

۲ . پاک کردن عدد :

کاربر با وارد کردن این دستور میتواند اعدادی که خودش به جدول اضافه کرده است را پاک کند. توجه کنید این دستور نمیتواند اعدادی که در ابتدا جزء جدول بودند را حذف کند. فرمت این دستور به این صورت میباشد که در یک خط دو ورودی وارد می شود که به ترتیب شماره سطر و شماره ستون جدول میباشد.

Delete

[row] [column]

۳ . راهنمایی :

در هر مرحله از اجرای بازی کاربر حداکثر سه مرتبه میتواند از این گزینه استفاده کند. در هر بار استفاده از این دستور سه خانه از خانه های حل نشده جدول به صورت رندوم انتخاب کرده و با اعدادی که جدول را به درستی حل میکنند، پر کنید.

Hint

3 numbers have been added to the table successfully .

در صورتی که کاربر از تمام راهنمایی های خودش استفاده کرده بود پیغام خطای مناسبی را نمایش دهید.

همچنین اگر تعداد خانه های حل نشده از 3 تا کمتر بود، پیام مناسبی در کنسول چاپ نمایید.

۴ . نمایش جدول :

با وارد کردن عبارت Show Table در کنسول، جدول سودوکو را به همراه تغییرات انجام شده تا آن لحظه در خروجی با فرمت خوانایی چاپ نمایید. برای نمایش خانه های خالی میتوانید از کاراکتر مشخصی استفاده کنید (* ، - ، ...)

۵ . خروج :

با اجرای این دستور از بازی خارج شده و طی یک پیام گزارشی از روند بازی کاربر ارائه دهید. این گزارش شامل تعداد خانه های حل شده و همچنین تعداد خانه های باقی مانده میباشد.

توجه ۲ : در هر مرحله از بازی اگر کاربر موفق به حل جدول شد پیغام مناسبی را به کاربر نمایش دهید و از برنامه خارج شوید . برای بررسی این موضوع دقت شود که برنامه شما باید متدی برای حل جدول داشته باشد و بتواند پس از حل جدول تشخیص دهد که آیا جدولی که کاربر حل کرده با جدول مورد انتظار یکسان است یا خیر (در صورتی که به طور دستی جدول را حل کنید و مقادیر مورد انتظار را در جدولی ذخیره کنید به شما نمره ای تعلق نمی گیرد و حتما برنامه شما باید توانایی حل جدول را داشته باشد)

قوانین بازی :

• طبق قوانین سودوکو کاربر نمی تواند دو عدد یکسان در یک بلاک(مربع های ۳ * ۳) ، یک ردیف و یا یک ستون وارد کند.

• کاربر نمی تواند در خانه هایی که به صورت پیش فرض در جدول تعیین شده اند را عددی اضافه یا آن را حذف کند.

• کاربر نمی تواند عددی خارج از بازه ۱ تا ۹ را برای پر کردن خانه ها، سطر و یا ستون انتخاب کند.

• اگر خانه ای توسط خود کاربر از قبل مقدار دهی شده باشد، در صورتی که دستور Add number برای آن خانه صدا زده شود نباید عدد آن خانه تغییر کند. در صورتی که بخواهد عدد آن خانه را تغییر دهد، ابتدا با استفاده از دستور Delete باید آن را حذف کند و سپس عدد جدید را اضافه کند.

• در حل این سوال می بایست از مفاهیم مقدماتی شی گرایی استفاده کنید.

بخش امتیازی :

برنامه شامل متدی برای ایجاد جدولی رندوم داشته باشد بطوریکه که کاربر در ابتدا و در منوی برنامه انتخاب کند که آیا می خواهد جدول پیش فرض برنامه را حل کند و یا می خواهد جدولی جدید برایش تولید شود . در صورت انتخاب گزینه دوم منویی برایش ظاهر شود که در آن سه سطح :

```
1.Easy
2.Normal
3.Hard
```

نوشته شده باشد و پس از آن که کاربر سطح سختی جدول را انتخاب کرد ، جدولی با آن سطح برایش تولید شود

توجه ۳ : دقت شود که جدول رندوم تولید شده باید قابل حل باشد و دارای جواب یکتا باشد .

توجه ۴ : سطوح بازی بر اساس تعداد خانه های پر شده اولیه طبقه بندی می شوند به این صورت که سطح easy شامل تعداد خانه های پر شده زیادی هست و سطح hard تعداد خانه های پر شده کمتری دارد

توجه ۵ : در صورتی که تنها اعداد جدول اولیه را با اعداد رندوم جا به جا کنید به شما نمره ای تعلق نمی گیرد و باید جایگاه های اعداد با جایگاه های اولیه جدول متفاوت باشد.

تمرین ۶ . احمد آقا و مستاجرینش

احمد آقا که دیگر پا به سن گذاشته است و نمیتواند مثل قدیم هر زمان که بخواهد دم در ساختمان هایش برود و از مستاجرین خود اجاره بگیرد، از شما میخواهد که برنامه ای برای او بنویسید که با کمترین تعداد بازدید از ساختمان، بتواند از بیشترین تعداد مستاجرین، اجاره را دریافت کند. هر یک از این بازدید های انجام شده از ساختمان در یک زمان مشخص اتفاق خواهد افتاد که این زمان مشخص، یک ساعت است. به عبارتی در هر ساعت میتوان از مجموعه ای از مستاجران، اجاره را دریافت کرد که این مجموعه ممکن است یک مجموعه تهی باشد یا به عبارتی از هیچ یک از مستاجران نتوان اجاره را دریافت کرد یا ممکن است شامل تمام مستاجران شود. بنابراین کاری که باید انجام دهید آن است که مجموعه ای از این ساعات را پیدا کنید که تعداد آن کمترین مقدار ممکن باشد، به طوری که از بیشترین تعداد مستاجرین بتوان اجاره را دریافت کرد.

برای آنکه شما بتوانید این برنامه را پیاده سازی کنید معلومات زیر در اختیار شما قرار خواهند گرفت:

۱ . بازهٔ زمانی ای که احمدآقا این امکان را دارد که به ساختمان برود و اجاره را از مستاجرینش دریافت نماید. نکته ای که وجود دارد آن است که خارج از این بازهٔ زمانی، امکان دریافت اجاره از مستاجران وجود نخواهد داشت.

۲ . بازه های زمانی ای که هر از یک مستاجران در خانهٔ خود حضور دارند.

نکته ۱ : اگر مستاجری در بازهٔ زمانی ای که احمدآقا امکان بازدید از ساختمان را دارد در ساختمان حضور نداشته باشد، امکان دریافت اجاره از او وجود نخواهد داشت. به عبارتی ممکن است که از تعدادی از مستاجران، نتوان اجاره را دریافت کرد .

توجه ۱ : به منظور رسیدن به هدف مساله یک تابع بنویسید که دو مقدار را به عنوان خروجی بازگرداند:

۱ . آرایه ای از اعداد صحیح که مقادیر موجود در آن، ساعاتی است که بازدید از ساختمان، در آن ها اتفاق خواهد افتاد.

۲ . بیشترین تعداد مستاجرانی که پس از بازدید از ساختمان، از آن ها اجاره دریافت خواهد شد.

ورودی

خط اول شامل دو عدد صحیح L و R است که با فاصله از یکدیگر جدا شده اند . این دو عدد به ترتیب بیانگر ابتدا و انتهای بازه ای میباشد که احمدآقا امکان بازدید از ساختمان را دارد.

خط دوم شامل عدد صحیح n است که نشان دهندهٔ تعداد مستاجرین موجود در ساختمان است.

در هر یک از n خط بعدی، دو عدد صحیح که با یک فاصله از یکدیگر جدا شده اند داده میشود. این دو عدد،به ترتیب بیانگر ابتدای بازه و انتهای بازهٔ حضور مستاجرین در خانه های خود میباشد.(L_i و R_i)

$$0 \le L \le R \le 10^9$$

$$0 \le L_i \le R_i \le 10^9$$

$$0 \le n \le 100$$

توجه ۲ : جهت ذخیرهٔ بازه های حضور مستاجران در ساختمان از آرایه دو بعدی استفاده کنید. همچنین هر عملیاتی که قرار است بر روی آرایه انجام شود به صورت یک تابع نوشته شود.

خروجی :

خروجی شامل دو خط است:

خط اول شامل تعداد مستاجرانی است که از آن ها اجاره دریافت شده است.

خط دوم شامل مجموعه ساعاتی است که بازدید از ساختمان در آن ها اتفاق افتاده است.

Example 1 :

Input :

```
3 7
3
3 6
1 2
7 8
```

Output :

```
2
6 8
```

دلیل : با توجه به خط نخست ورودی، متوجه میشویم که احمد آقا در بین ساعات ۳ تا ۷ میتواند از ساختمان بازدید کند. نکته آن است که در ابتدا و انتهای بازهٔ ذکر شده نیز امکان بازدید وجود خواهد داشت. پس از خط نخست، در خط دوم تعداد مستاجران آمده است که در این نمونه، این مقدار، ۳ میباشد. در هر یک از ۳ خط بعدی، بازهٔ حضور هر یک از مستاجران به عنوان ورودی داده شده است. به طوری که مستاجر اول در بین ساعات ۳ تا ۶ در خانه حضور دارد. مستاجر بعدی در بین ساعات ۱ تا ۲ و مستاجر آخر در بین ساعات ۷ تا ۸ در خانهٔ خود حضور خواهند داشت.

نکته ای که در مورد این بازه های زمانی وجود دارد، آن است که در ابتدا و انتهای بازه نیز مستاجران در خانه حضور خواهند داشت. به عنوان مثال مستاجر اول علاوه بر این که بین ساعات ۱ و ۲ در خانه حضور دارد، در خود ساعات ۱ و ۲ نیز در خانه حضور خواهد داشت.

از آنجا که مستاجر دوم به طور کامل در بازه ای که احمدآقا امکان بازدید از ساختمان را دارد، در خانه حضور نخواهد داشت، بنابراین از این مستاجر امکان دریافت اجاره وجود نخواهد داشت. در حالی که میتوان از دو مستاجر دیگر اجاره را دریافت کرد. بنابراین حداکثر تعداد مستاجرانی که میتوان از آن ها اجاره را دریافت کرد مقدار ۲ است که این مقدار در خط اول خروجی آمده است. در خط دوم خروجی، مجموعه ساعاتی که این بازدید ها اتفاق افتاده است آمده است. این ساعات باید به گونه انتخاب شوند که تعداد آن ها کمترین تعداد ممکن شود. در ساعت ۶ ، میتوان از مستاجر اول اجاره را گرفت و در ساعت ۸ ، از مستاجر آخر میتوان این مقدار را دریافت کرد.

Example 2 :

Input :

```
4 10
10
3 3
1 3
2 4
5 8
4 5
8 10
10 10
4 7
4 10
11 14
```

Output :

```
7
4 8 10
```

Example 3 :

Input :

```
6 10
5
1 3
4 7
6 8
7 8
8 9
```

Output :

```
4
7 9
```

تمرین ۷ . بورس پیشرفته !

بعنوان یک آنالیزور بازار بورس و سهام قصد داریم تا وضعیت خرید و فروش یک شرکت را تحلیل کنیم .

برای این کار می بایست کلاسی به نام Stocks پیاده سازی کنیم بطوریکه فیلد های زیر را داشته باشد :

۱ . TotalCash : موجودی شرکت برای خرید سهم

۲ . StockValues[] : ارزش سهم مد نظر در یک بازه خاص که بصورت آرایه ای از اعداد صحیح می باشد - در واقع هر المان آرایه بیانگر ارزش سهام در روز i ام می باشد (تضمین می شود که طول این بازه بیشتر از ۱۰۰ نخواهد بود)

این کلاس شامل توابع زیر می باشد :

۱ . تابع BuyStockMethods :

این تابع بعنوان ورودی آرایه ای از اعداد صحیح که شامل مقادیری است که شرکت در هر روز دقیقا به همان میزان می تواند خرید ثبت کند را دریافت می کند

در خروجی تمام روش های خرید یک سهم را برمیگرداند - به مثال توجه کنید :

Example 1 :

در این مثال موجودی شرکت ۴ میلیون دلار در نظر گرفته شده است

```
TotalCash = 4
```

و به تابع مقادیر زیر بعنوان ورودی داده می شوند :

```
dailyBuys = { 1 , 2 , 3 }
```

خروجی تابع شامل تمام حالت های تقسیم موجودی با توجه به آرایه ورودی می باشد :

```
{ 1 , 1 , 1 , 1 } , { 1 , 1 , 2 } , { 2 , 2 } , {1,3}
```

نکته ۱ : از آنجایی که برای هر کدام از روش ها ترتیب های متفاوتی وجود دارد ، هر کدام از ترتیب ها را برگردانید تفاوتی ندارد و درست است :

```
{ 1 , 1 , 2 } = { 1 , 2 , 1 } = { 2 , 1 , 1 }
```

نکته ۲ : همچنین ترتیب کلی روش ها نیز مهم نیست :

```
{ 1 , 1 , 2 ) , { 2 , 2 } = { 2 , 2 } , { 1 , 1 , 2 }
```

Example 2 :

در این مثال موجودی شرکت ۱۰ میلیون دلار در نظر گرفته شده است

```
TotalCash = 10
```

و به تابع مقادیر زیر بعنوان ورودی داده می شوند :

```
dailyBuys = { 2 , 5 , 3 , 6 }
```

خروجی تابع شامل تمام حالت های تقسیم موجودی با توجه به آرایه ورودی می باشد :

```
{ 2 , 2 , 2 , 2 , 2 }  , { 2 , 2 , 3 , 3 } ,
{ 2 , 2 , 6 } , { 2 , 3 , 5 } , { 5 , 5 }
```

۲ . تابع BestPeriods :

در این تابع قصد داریم با توجه به ارزش سهام در یک بازه خاص بیشترین سود ممکن را با انتخاب درست بازه های خرید و فروش دریافت کنیم

در واقع با توجه به فیلد StockValues و تشخیص بهترین زمان برای خرید و فروش سهم ، حداکثر سودی دریافتی را می یابیم

بدیهی است اگر ارزش سهم همواره کاهش یابد مقدار سود دریافتی صفر خواهد بود

به مثال توجه کنید :

Example 1 :

در این مثال آرایه StockValues به صورت زیر است :

```
StockValues[] = {100,180,260,310,40,535,695}
```

یکی از راه های ممکن برای اینکه بیشترین سود را دریافت کنیم (ممکن است در مثالی چندین راه برای دریافت بیشترین سود داشته باشیم) به صورت زیر است :

```
( 0 , 3 ) , ( 4 , 6 )
```

به این معنا که در روز صفرم سهم را خریده و سپس در روز سوم آن را بفروشیم . سپس در روز چهارم آن را خریداری کرده و در روز ششم آن را بفروشیم که در نهایت با توجه به مقادیر آرایه حداکثر سود بدست می آید :

```
( StockValues[3] - StockValues[0] ) +
( StockValues[6] - StockValues[4] ) =
( 310 - 100 ) + ( 695 - 40 ) = 865
```

و تابع مقدار 865 را بعنوان خروجی برمیگرداند

Example 2 :

در این مثال آرایه StockValues به صورت زیر است :

```
StockValues[] = {4,2,2,2,4}
```

یکی از راه های ممکن برای اینکه بیشترین سود را دریافت کنیم (ممکن است در مثالی چندین راه برای دریافت بیشترین سود داشته باشیم) به صورت زیر است :

```
( 3 , 4 )
```

به این معنا که در روز سوم سهم را خریده و سپس در روز چهارم آن را بفروشیم که در نهایت با توجه به مقادیر آرایه حداکثر سود بدست می آید :

```
StockValues[4] - StockValues[3]  = ( 4 - 2 ) = 2
```

و تابع مقدار 2 را بعنوان خروجی برمیگرداند

۳ . تابع ProfitSpan :

برای المان i ام از آرایه StockValues ،عدد S_i بصورت زیر تعریف می شود :

بیشترین تعداد روز متوالی تا قبل از روز i ام ، بطوریکه ارزش سهام در این روز - روز i ام - بیشتر یا مساوی ارزش سهام در روز های قبل از خود باشد

به بیان دیگر اگر بازه L_i را روز های متوالی تا قبل روز i ام در نظر بگیریم بطوریکه در تمامی این روز ها ارزش سهام از روز i ام کمتر یا مساوی باشد ، S_i برابر با طول این بازه خواهد بود

در این تابع قصد داریم مقدار S_i را برای تمامی المان های آرایه StockValues بدست آورده و در قالب یک آرایه برگردانیم

به مثال توجه کنید :

Example 1 :

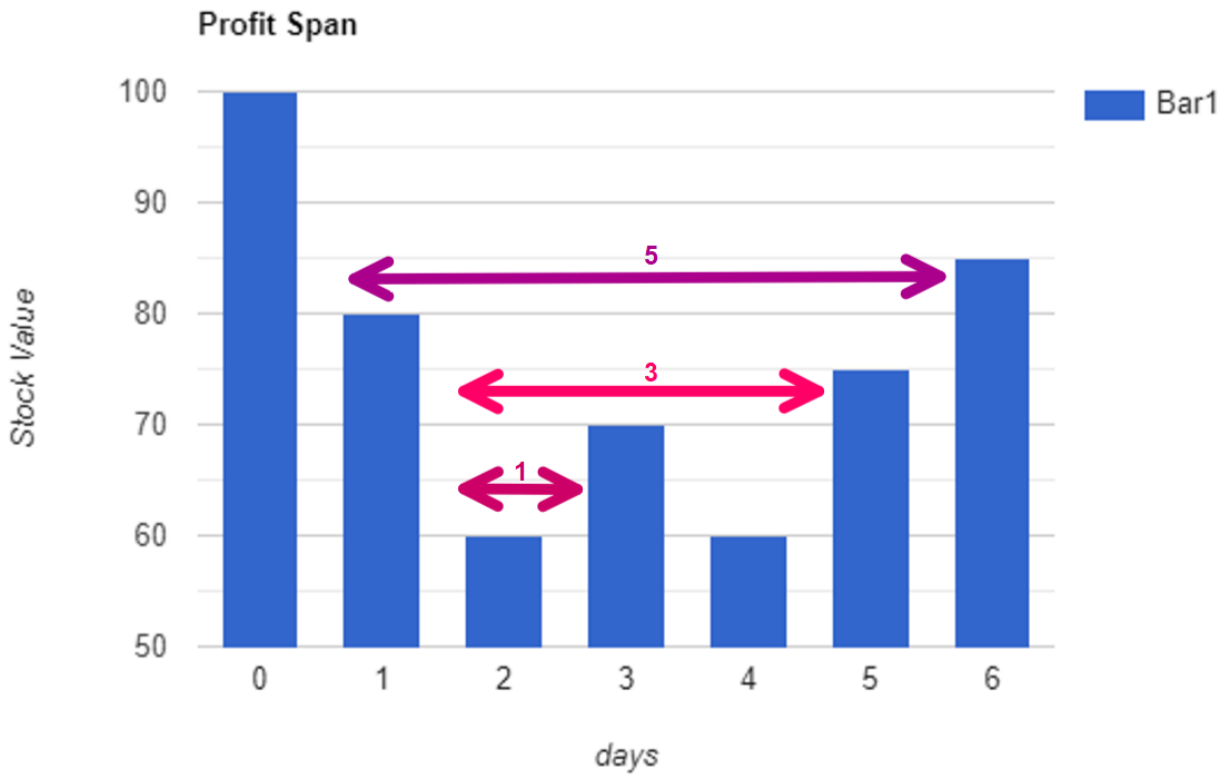
در این مثال آرایه StockValues به صورت زیر است :

```
StockValues[] = {100,80,60,70,60,75,85}
```

خروجی تابع بصورت زیر خواهد بود :

```
[0,0,0,1,0,3,5]
```

توضیحات خروجی در صفحه بعد آورده شده :



المان های ۰ تا ۲ : از آنجایی که نمودار تا قبل از آنها نزولی است S_i برابر با 0 خواهد بود

المان ۳ ام : تنها در روز قبل از خود مقدار سهام کمتر است به همین دلیل S_3 برابر با 1 خواهد بود

المان ۴ ام : چون روز قبل ارزش سهم بیشتر از آن است پس S_4 برابر با 0 خواهد بود

المان ۵ ام : از المان دوم تا قبل از المان ۵ ام تمامی ارزش های سهام از این روز کمتر بوده به همین دلیل بازه L_i برابر با :

```
2 : 5 –> ( 2 , 3 , 4 )
```

خواهد بود که طول آن برابر ۳ می باشد

المان ۶ ام : بجز المان صفرم تمامی مقادیر آرایه از آن کمتر بوده و بازه L_i بصورت زیر خواهد بود :

```
1 : 6 –> ( 1 , 2 , 3 , 4 , 5 )
```

موفق باشید :)