

پیاده‌سازی - گزارش کار

• محدودیت زمان: ۱ ثانیه

• محدودیت حافظه: ۲۵۶ مگابایت

احمد به تازگی در شرکتی که نخواسته اسم آن فاش شود استخدام شده است. وی در بخش بسته‌بندی کار می‌کند و گزارش به این شکل است:

۱. از یک روز قبل، شرکت به او تعداد بطری‌ها و همچنین ظرفیت هر کدام از آن‌ها را می‌دهد.
۲. سپس به او یک عدد k داده می‌شود و از او خواسته می‌شود k لیتر مایع محربانه درون بطری‌ها بریزد.
۳. در نهایت در همان روز یک گزارش بفرستد و بگویید که می‌توان این حجم از مایع را در این بطری‌ها ریخت یا خیر، و در صورتی که بتوان ریخت، باید به شرکت برود و مایع محربانه را در بطری‌ها بریزد.

امروز احمد ایمیلی دریافت کرده که در آن گفته شده که باید از همین شنبه گزارش را شروع کند و به شرکت بباید. همچنین تعداد و ظرفیت بطری‌ها و مقدار حجم مایع محربانه نیز در ایمیل به او داده شده و این جمله نیز ذکر شده: "در صورتی که تا پایان امشب گزارش را نفرستید، اخراج می‌شوید!"

حال احمد در این دوراهی قرار گرفته که مسابقه امروز استپتریپ را بدهد یا گزارشش را برای شرکت بنویسد. از آنجایی که احمد اهل رقابت است، ترجیح می‌دهد مسابقه را بدهد و تعداد و ظرفیت بطری‌ها و حجم مایع محربانه را به شما می‌دهد تا شما برایش گزارش را بنویسید.

شما باید با دریافت تعداد بطری‌ها و ظرفیت هر کدام و مقدار حجم مایع، بگویید می‌توان این حجم از مایع را در بطری‌ها ریخت یا نه، همچنین این را می‌دانیم که هر بطری در ابتدا **خالی** می‌باشد و حداقل به میزان ظرفیتش می‌تواند مایع را ذخیره کند.

توجه کنید که لزومی ندارد بطری‌ها به طور کامل پر شوند.

ورودی

در خط اول ورودی به شما دو عدد n و k داده می‌شود که به ترتیب تعداد بطری‌ها و حجم مایع محربانه به لیتر

می‌باشد. در n خط بعدی از ورودی، در خط i ام عدد c_i می‌آید که بیانگر ظرفیت بطری i ام به لیتر می‌باشد.

$$1 \leq n \leq 100$$

$$1 \leq k \leq 100\,000$$

$$1 \leq c_i \leq 1\,000$$

خروجی

در صورتی که می‌توان این حجم از مایع را در بطری‌ها ریخت YES و در غیراین‌صورت NO را چاپ کنید.

مثال

ورودی نمونه ۱

3 3
1
2
1

خروجی نمونه ۱

YES

احمد می‌تواند ۱ لیتر از مایع را در بطری شماره ۱ و ۲ لیتر باقیمانده را در بطری شماره ۲ بریزد.

ورودی نمونه ۲

2 5
3

۱

خروجی نمونه ۲

NO

پیاده‌سازی - کتیبه تاریخی

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

احمد به دلیل خستگی زیاد، با دوستان خود به کوههای دور دست رفته است. وی که به گشت‌وگذار علاقه‌مند است، شروع به گردش در آن مکان می‌کند.

احمد ناگهان یک کتیبه سنگی را می‌بیند و به سمت آن می‌رود. در کنار آن کتیبه یک قفل تاریخی قرار دارد. در بالای کتیبه یک نوشته با این متن وجود دارد: "رمز قفل را حدس بزنید تا جاودانه شوید!" همچنین در کنار آن یک دستورالعمل و یک جدول $m \times n$ وجود دارد که هر خانه آن . و یا * است. همچنین سطرهای این جدول از بالا به پایین به ترتیب با اعداد ۱ تا n و ستون‌های آن از چپ به راست به ترتیب با اعداد ۱ تا m شماره‌گذاری شده‌اند و منظور از خانه (j, i) ، خانه‌ای است که در تقاطع سطر i ام و ستون j ام جدول وجود دارد.

در دستورالعمل گفته شده که شما در جدولی که روی کتیبه است باید تعداد الگوهای L مانند را بشمارید و رمز قفل برابر با تعداد این الگوها می‌باشد. همچنین فقط یک فرصت برای وارد کردن رمز قفل دارید و در صورت اشتباه بودن رمز، دیگر نمی‌توانید جاودانه شوید.

یک الگوی L مانند در جدول مت Shank از k خانه افقی متواالی شامل * و $2k$ خانه عمودی متواالی شامل * می‌باشد که پایین‌ترین خانه‌ی تکه عمودی و چپ‌ترین خانه‌ی تکه افقی با یکدیگر مشترک می‌باشند و یک شکل شبیه حرف L انگلیسی تشکیل می‌دهند (یعنی دوران‌های دیگر حرف L شمرده نمی‌شوند).

همچنین ممکن است چند L مختلف با یکدیگر دارای اشتراک باشند.

حال احمد به دلیل اینکه هیجان‌زده شده، نمی‌تواند تمرکز کند و برای همین جدول روی کتیبه را به شما می‌دهد تا شما برایش الگوهای L مانند را بشمارید و به او بگویید.

ورودی

در خط اول ورودی دو عدد n و m می‌آیند که به ترتیب بیانگر تعداد سطر و ستون جدول می‌باشند.

در n امین خط از n خط بعدی، یک رشته به طول m متشكل از \cdot و $*$ داده می‌شود که n امین عنصر آن برابر با مقدار خانه واقع در تقاطع سطر n ام و ستون n ام می‌باشد.

$$1 \leq n, m \leq 100$$

خروجی

در تنها خط خروجی، تعداد الگوهای L مانند را خروجی دهید.

مثال

ورودی نمونه ۱

```
4 4
*...
*...
*...
*****

```

خروجی نمونه ۱

1

تنها یک L وجود دارد که شامل خانه‌های زیر می‌باشد:

$(1,1), (2,1), (3,1), (4,1), (4,2)$

ورودی نمونه ۲

```
5 3
*..
*..

```


خروجی نمونه ۲

2

دو L وجود دارند که شامل خانه‌های زیر می‌باشند:

(1, 1), (2, 1), (3, 1), (4, 1), (4, 2)

(2, 1), (3, 1), (4, 1), (5, 1), (5, 2)

ورودی نمونه ۳

4 4
....
. * ..
. * ..
. ** .

خروجی نمونه ۳

0

ورودی نمونه ۴

4 3
. * .
. * .
. * .

خروجی نمونه ۴

۱

یک L وجود دارد که شامل خانه‌های زیر می‌باشد:

(1, 2), (2, 2), (3, 2), (4, 2), (4, 3)

پیادهسازی - ازدواج‌های لحظه‌ای

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

احمد عمومی به اسم هوشنگ دارد که شخصیت مرموز او همواره برای احمد جالب توجه بوده است.

آقا هوشنگ n دختر دارد که به ترتیب سن از کوچک به بزرگ، با ۱ تا n شماره‌گذاری شده‌اند. دختران آقا هوشنگ دوست دارند هر چه سریع‌تر ازدواج کنند.

پدر آن‌ها مشکلی با این ویژگی دخترانش ندارد ولی از آنجایی که می‌خواهد متفاوت باشد، شرطی برای ازدواج دخترانش گذاشته که یک دختر تنها زمانی می‌تواند ازدواج کند که همه دخترهای کوچک‌تر از او، ازدواج کرده باشند.

احمد می‌داند که n خواستگار می‌خواهند به خواستگاری دختران آقا هوشنگ بیایند و هیچ دو تایی از آن‌ها به دختر یکسانی علاقه‌مند نیستند. خواستگار i ام به خواستگاری دختر a_i ام آقا هوشنگ می‌رود.

از آنجایی که خانه آقا هوشنگ خیلی کوچک است، در هر روز فقط یک خواستگار می‌تواند بیاید و از یکی از دخترانش خواستگاری کند و جواب بله را بگیرد. اما این پایان کار نیست و هر دختری باید تا ازدواج کردن همه دختران کوچک‌تر از خود صبر کند (یعنی زمانی که همه دختران کوچک‌تر از او ازدواج کردند، آن دختر ازدواج می‌کند).

حال احمد می‌خواهد بداند که هر کدام از دختر عموهایش در چه روزی ازدواج می‌کنند تا برای آن‌ها کادوی مناسب تدارک بییند. از آنجایی که آقا هوشنگ خود عاقد است، هر کدام از ازدواج‌ها تنها یک لحظه طول می‌کشد و بنابراین تعداد ازدواج‌هایی که در یک روز صورت می‌گیرد، محدودیتی ندارد.

ورودی

در خط اول ورودی عدد n آمده است که نشان‌دهنده تعداد دختر عموهای احمد می‌باشد. در i امین خط از n خط بعدی عدد a_i آمده است که نشان‌دهنده سن دختری است که در روز i ام از او خواستگاری می‌شود.

$$1 \leq n \leq 100\,000$$

$$1 \leq a_i \leq n$$

خروجی

در n آمین خط از n خط خروجی چاپ کنید که n آمین کوچکترین دختر آقا هوشنگ در روز چندم ازدواج می‌کند.

مثال

ورودی نمونه ۱

3
1
2
3

خروجی نمونه ۱

1
2
3

در روز اول برای دختر اول آقا هوشنگ خواستگار می‌آید و ازدواج می‌کند. در روز دوم برای دختر دوم خواستگار می‌آید و چون دختر اول ازدواج کرده است، دختر دوم در همین روز ازدواج می‌کند. در روز سوم نیز برای دختر سوم خواستگار می‌آید و چون دو دختر دیگر قبل ازدواج کرده‌اند، دختر سوم در همین روز ازدواج می‌کند.

ورودی نمونه ۲

4
3

1
2
4

خروجی نمونه ۲

2
3
3
4

در روز اول برای دختر سوم آقا هوشنگ خواستگار می‌آید ولی چون هنوز دختر اول و دوم ازدواج نکرده‌اند، ازدواجی صورت نمی‌گیرد. در روز دوم برای دختر اول خواستگار می‌آید و او در همان روز ازدواج می‌کند. در روز سوم برای دختر دوم خواستگار می‌آید و او در همان روز ازدواج می‌کند، چون دختر اول پیش از آن ازدواج کرده است. همچنین دختر سوم هم که پیش از این منتظر ازدواج خواهران کوچکترش بود، در همین روز بعد از دختر دوم ازدواج می‌کند. در روز چهارم برای دختر چهارم خواستگار می‌آید و چون خواهران کوچکتر او ازدواج کرده‌اند، او هم در همین روز ازدواج می‌کند.

پیاده‌سازی - مسابقات سنگین

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

احمد مسئول برگزاری یک سری مسابقات سنگین شده است. او از این کار جدید خیلی خوشحال است ولی برگزاری چنین مسابقاتی خیلی سخت است و نیاز به یک برنامه‌نویس خبره برای نوشتن سیستم مدیریت آن دارد، برای همین او از شما می‌خواهد تا سیستمی طراحی کنید که این مسابقات را مدیریت کند.

مسابقات به این شکل برگزار می‌شود که در هر لحظه تعدادی بازیکن در مسابقات وجود دارند که می‌خواهند با یکدیگر بازی کنند. همچنین تعدادی درخواست بازی در هر زمان وجود دارد که در پایین توضیح داده می‌شود.

هر بازیکن یک قدرت (s_i) و یک جنبه، (g_i) دارد. اگر کسی تعداد بازی‌هایی که باخته است بیشتر از جنبه‌اش شود، از مسابقات انصراف می‌دهد. همچنین اگر کسی تقلب کند، از مسابقات حذف خواهد شد و به تبع آن، درخواست‌هایش نیز نادیده گرفته می‌شوند.

بازی‌ها تنها با توافق طرفین انجام می‌شود، یعنی زمانی یک بازی شکل می‌گیرد که هر دو طرف راضی باشند که با حریفی با ویژگی‌های فعلی رقابت کنند. نحوه ایجاد بازی‌ها هم بدین‌گونه است که بازیکن‌ها درخواست بازی می‌دهند و اگر شرایط لازم برای آن درخواست را داشتند، درخواستشان اجرا می‌شود و در غیر این صورت نادیده گرفته می‌شود.

اجرا شدن درخواست‌های بازی به شکل زیر است:

1. در هر مرحله یک مجموعه از درخواست‌های بازی، از افراد مختلف وجود دارد.
2. وقتی درخواست جدیدی ثبت می‌شود، بررسی می‌شود که آیا با درخواست‌های قبلی مطابقت دارد یا نه. اگر مطابقت نداشت به مجموعه اضافه می‌شود.
3. اگر درخواست جدید با چند مورد از درخواست‌های صفت انتظار مطابقت داشت، اولویت انتخاب به ترتیب با موارد زیر است (در صورت تساوی بازیکنان در یک زمینه به اولویت بعدی نگاه می‌کنیم):
 - بازیکن درخواست‌کننده امتیاز کمتری داشته باشد. امتیاز آن بازیکن در زمان ایجاد درخواست مدد

نظر است و امتیاز فعلی آن اهمیتی ندارد.

- بازیکن درخواست‌کننده جنبه بیشتری داشته باشد.

- درخواست مورد نظر قدیمی‌تر باشد، یعنی در زمان زودتری ایجاد شده باشد.

می‌دانیم در بازی بازیکن‌های x و y ، اگر قدرت بازیکن x را g_x و قدرت بازیکن y را g_y در نظر بگیریم، بازیکنی که زودتر درخواست بازی را داده است می‌برد، اگر و تنها اگر، باقی‌مانده تقسیم عدد $g_y \times g_x$ بر $g_y + g_x$ بزرگتر از $|g_x - g_y|$ باشد. در انتها نیز برنده بازی به اندازه قدرت بازیکن مقابله، امتیاز دریافت می‌کند و تغییری در امتیاز بازنده ایجاد نمی‌شود.

می‌دانیم هر دقیقه‌ای که می‌گذرد، یک اتفاق جدید می‌افتد و نتیجه آن بر مسابقات در همان دقیقه اعمال می‌شود.
در هر دقیقه از بازی یکی از اتفاقات زیر رخ می‌دهد:

۱. بازیکنی به اسم s ، با قدرت g و جنبه k به بازی اضافه می‌شود. دقت کنید که امتیاز هر بازیکن زمان اضافه شدن به بازی صفر است (اسم‌های بازیکن‌ها متمایز است).

۲. کمیته مسابقات متوجه تقلب بازیکنی با نام s می‌شود.

۳. بازیکنی با نام s درخواست بازی عمومی می‌کند، یعنی اگر شخص دیگری درخواستی داده که s ، شامل شرایط آن می‌شود، با او بازی می‌کند.

۴. بازیکنی با نام s درخواست بازی با بازیکنی که امتیاز آن حداقل l و حداقل m است را می‌کند. برای پیگیری این درخواست توسط کمیته مسابقات، این بازیکن باید حداقل n امتیاز داشته باشد و پس از اجرای این درخواست هم o امتیاز از این بازیکن کم می‌شود. (دقت کنید موقعی که درخواست پذیرفته می‌شود، امتیاز کسر نمی‌شود و زمانی که بازی انجام می‌شود، امتیاز موردنظر کسر می‌شود).

۵. بازیکنی با نام s درخواست بازی اختصاصی با بازیکنی به اسم t را می‌کند. برای اینکه درخواست بازیکن پذیرفته شود، باید حداقل p امتیاز داشته باشد و پس از اجرای این درخواست (انجام بازی)، q امتیاز از بازیکن کسر می‌شود. می‌دانیم که وقتی این درخواست صورت می‌گیرد، بازیکنی با نام t حتماً در مسابقات حاضر است.

۶. جدول رده‌بندی را تا لحظه فعلی چاپ کنید، بدین شکل که ابتدا در یک خط عبارت scoreboard: را چاپ کنید. سپس بازیکن‌ها بر اساس اولویت‌های زیر مرتب کنید و اسم آن‌ها به ترتیب در خطوط جداگانه خروجی دهید:

- بازیکن امتیاز بالاتری داشته باشد.

- قدرت بازیکن کمتر باشد.
- جنبه بازیکن بیشتر باشد.
- تعداد درخواست‌های مورد قبول بازیکن تا بدین‌جا کمتر باشد. (اجرا شدن درخواست‌ها برایمان اهمیتی ندارد و فقط وارد شدن آن‌ها به صف انتظار برایمان مهم است)
- اسم بازیکن از نظر کتابخانه‌ای کوچک‌تر باشد. (برای آشنایی با ترتیب کتابخانه‌ای می‌توانید به این لینک مراجعه کنید)

به موارد زیر هم در هنگام پیاده‌سازی توجه کنید:

- اگر کسی از مسابقات انصراف دهد یا حذف شود، دیگر در جدول رده‌بندی چاپ نمی‌شود و اجازه بازی جدید ندارد. همچنین اگر درخواست جدیدی بدهد، نادیده گرفته می‌شود و درخواست‌های قبلی او نیز از صفات انتظار حذف خواهند شد.
- اگر یکی از درخواست‌ها شرایط مدنظر کمیته را نداشت لازم نیست چیزی چاپ کنید و صرفا آن درخواست حذف می‌شود.
- نام افراد حاضر در مسابقه با هم متمایز است.

ورودی

با ظاهر شدن عبارت `start` در ورودی مسابقات شروع می‌شود. پس از شروع مسابقات، هر خط نشان‌دهنده بروز یکی از ۶ اتفاق ممکن در صورت سوال است. قالب اتفاقات مختلف به صورت زیر است (نامین قالب مربوط به نامین نوع اتفاق در بازی است):

۱. `add $ss g $k`
۲. `$ss cheats`
۳. `$ss competes everybody`
۴. `$ss competes between 1 r`
۵. `$ss competes t`
۶. `print scoreboard`

تضمين می‌شود هر خط از ورودی دقیقا با یکی از قالب‌های ارائه شده همخوانی دارد. همچنین اسمی بازیکنان

تنها از حروف کوچک انگلیسی تشکیل شده است و همه اعداد نیز از صفر تا هزار هستند.

با ظاهر شدن عبارت `end` ورودی پایان می‌یابد.

تعداد خطوط ورودی حداکثر هزار تاست.

خروجی

به ازای هر بار درخواست شماره `6`، جدول رده‌بندی را به صورت کامل چاپ کنید.

مثال

ورودی نمونه ۱

```
start
print scoreboard
add ali 10 1
add ahmad 0 0
print scoreboard
ahmad competes everybody
ahmad competes everybody
ali competes everybody
ali competes everybody
add erfan 100 0
erfan competes everybody
print scoreboard
erfan competes between 0 100
print scoreboard
ahmad competes between 0 100
print scoreboard
ahmad competes everybody
print scoreboard
end
```

خروجی نمونه ۱

```
scoreboard:  
scoreboard:  
ahmad  
ali  
scoreboard:  
erfan  
ali
```

در ابتدا که هیچ شرکت‌کننده‌ای وجود ندارد.

پس از آن دو بازیکن به نام‌های علی و احمد به مسابقات اضافه می‌شوند که قدرت آن‌ها به ترتیب برابر ده و صفر می‌باشد و لذا احمد که قدرت کمتری دارد، در رده‌بندی بالاتر است.

پس از آن احمد دو بار درخواست رقابت عمومی می‌دهد و درخواست‌هایش پذیرفته شده و داخل صف انتظار قرار می‌گیرند. سپس علی درخواست رقابت عمومی می‌دهد و این درخواست او با درخواست اول احمد مطابقت پیدا کرد و آن دو به رقابت پرداختند. در اینجا چون احمد در ابتدا درخواست داده بود و شرط گفته شده در صورت سوال برقرار نبود، علی برنده بازی می‌شود و چون جنبه احمد صفر است و تعداد باخت‌هایش بیشتر از جنبه‌اش شده است، از مسابقات انصراف می‌دهد. البته با این برد، علی امتیاز خاصی به دست نمی‌آورد چون قدرت احمد برابر صفر بوده است. پس از آن علی یک بار دیگر درخواست بازی ارائه می‌دهد ولی چون درخواستی در صف انتظار وجود ندارد (درخواست احمد پس از حذف او از صف حذف شده است)، باید منتظر درخواست دیگری بماند.

حال بازیکنی جدید به نام عرفان و با قدرت صد به بازی اضافه می‌شود و درخواست یک بازی عمومی را می‌کند و چون درخواستش با درخواست علی مطابقت دارد، این دو وارد بازی می‌شوند. چون علی درخواست‌کننده اول بوده و همچنین شرط گفته شده در صورت سوال برقرار نیست، پس عرفان برنده بازی می‌شود و ده امتیاز به دست

می‌آورد و در جدول رده‌بندی در صدر قرار می‌گیرد.

حال عرفان یک درخواست بازی جدید ثبت می‌کند و چون امتیاز اون بیشتر مساوی پنج است، درخواستش به صف انتظار می‌رود ولی از آن‌جایی که پس از او فقط احمد درخواست می‌دهد و درخواست‌های او توسط کمیته مسابقات نادیده گرفته می‌شوند، دیگر بازی صورت نمی‌گیرد و جدول رده‌بندی مسابقات تا انتهای ثابت می‌ماند.

جاوا - نقاط دور از دسترسی

پرهام در تولید یک نرمافزار مسیریابی تصمیم گرفته است برای ذخیره‌سازی فاصله بین شهرها از یک ماتریس استفاده کند. او در کلاس *DistanceMap**، ماتریس *map** را تعریف کرده که سطرهای آن شماره‌ی شهر مبدأ، ستون‌های آن شماره‌ی شهر مقصد و درایه‌ی j, i آن فاصله بین دو شهر متناظر را نشان می‌دهد.

شاید راه حل پرهام برای ذخیره فاصله‌ها، چندان مناسب نباشد اما به او کمک کنید در تکمیل نرمافزارش راه مناسبی برای یافتن دو شهری که بیشترین فاصله را از یکدیگر نسبت به باقی شهرها دارند پیدا کند.

فایل [source](#) را دانلود کنید.

متند `main` در کلاس `MaxDistanceCalculator` را به گونه‌ای پیاده‌سازی کنید که با توجه به محتویات ماتریس `map` در کلاس `DistanceMap`، شماره‌ی دو شهری را که بیشترین فاصله را از یکدیگر دارند به دست آورده و در خروجی استاندارد چاپ کند.

به نکات زیر توجه کنید:

- سایز ماتریس `map` حداقل ۵ است.
- شماره‌ی شهرها از یک شروع می‌شود.
- فاصله‌ی هر شهر با خودش صفر است اما فاصله‌ی هر دو شهر دلخواه نیز می‌تواند صفر باشد.
- ماتریس `map` متقارن (مربعی) است و درایه‌ی j, i برابر i, j می‌باشد. در خروجی حالتی را چاپ کنید که شماره‌ی شهر مبدا بزرگتر یا مساوی شماره‌ی شهر مقصد است.
- ممکن است بیش از یک حالت پیدا شود پس جوابی را چاپ کنید که شماره‌ی مبدا و مقصد کمترین باشد.

مثال اول

محتویات ماتریس `map` :

00
00

خروجی:

1,1

مثال دوم

محتویات ماتریس : map

051
508
180

خروجی:

3,2

آن چه که باید آپلود کنید

یک فایل زیپ که وقتی آن را باز می‌کنیم فقط فایل MaxDistanceCalculator.java را ببینیم.

جاوا - اسپم

یکی از چالش‌های امروزی در زمینه‌ی شبکه‌های اجتماعی، شناسایی پیام‌های اسپم (*spam*) می‌باشد. معمولاً پیام‌ها از نظر اسپم بودن در چهار دسته زیر طبقه‌بندی می‌شوند:

- دسته **:Not Spam [NOT_SPAM]**
 - پیام‌هایی که اسپم طبقه‌بندی نمی‌شوند.
- دسته **:Invalid Sender [INVALID_SENDER]**
 - پیام‌هایی که شناسه فرستنده آن‌ها تنها از اعداد تشکیل شده است.
- دسته **:Invalid Content [INVALID_CONTENT]**
 - پیام‌هایی که در بدنه آن‌ها تعداد کاراکترهای غیرحرف، غیرعدد و غیرفاصله بیش از نصف طول پیام باشد و در آن حداقل یک بار زیرنویسی *spam* تکرار شده باشد.
- دسته **:Fully Invalid [FULLY_INVALID]**
 - پیام‌هایی که هم فرستنده و هم بدنه‌ی نامعتبر داشته باشند.

بسته **spam** را دانلود کنید.

متدهای `detectSpams` در کلاس `SpamDetector` را به گونه‌ای پیاده‌سازی کنید که با دریافت آرایه‌ای از آرایه‌های از `SpamType` ها، آرایه‌ای از `Message` ها را برگرداند که هر عنصر آن نوع پیام متناظر در آرایه‌ی `messages` را مشخص می‌کند. به عبارتی اگر عنصر اول آرایه‌ی `messages` اسپم نباشد باید عنصر اول آرایه‌ی برگشتی از نوع `NOT_SPAM` باشد.

برای مثال با اجرای متدهای `main` در کلاس `Main`، باید خروجی زیر چاپ شود:

```
[INVALID_SENDER, INVALID_CONTENT]
```

آن چه که باید آپلود کنید

یک فایل زیپ که وقتی آن را باز می‌کنیم فقط فایل `SpamDetector.java` را ببینیم.

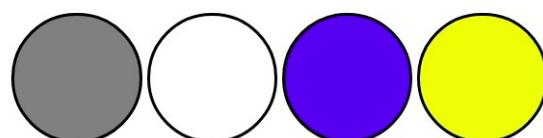
فرانت‌اند - Background Color Switch

مهارت‌های لازم:

- آشنایی با JS

در این سوال قصد داریم برای دکمه‌هایی که از قبل تعریف شده اند عملکردی اضافه کنیم تا با کلیک بر روی هر کدام از آنها رنگ پس زمینه به همان رنگ دکمه تغییر پیدا کند. ظاهر کلی برنامه بدین صورت است:

Color Scheme Switcher



پروژه اولیه

پروژه اولیه را از [اینجا](#) دانلود کنید. ساختار فایل‌های این پروژه به صورت زیر است.

```
background-color-switch
├── index.html
└── styles.css
```

جزئیات

تغییرات لازم را در فایل `app.js` اعمال کنید.

نکات

- توجه کنید که داوری خودکار بر مبنای نام کلاس های انجام می شود.
- پروژه را با ساختار زیر ارسال کنید.

```
[your-zip-file-name].zip
└── app.js
```

فرانت‌اند - Autocomplete

مهارت‌های لازم:

- آشنایی با JS

می‌خواهیم با استفاده از جاوااسکریپت بر اساس یک لیستی از برخی استان‌های ایران، یک Autocomplete مطابق با شکل زیر پیاده سازی کنیم.



پروژه اولیه

پروژه اولیه را از [لينجا](#) دانلود کنید. ساختار فایل‌های این پروژه به صورت زیر است.

```
autocomplete
├── app.js
└── index.html
└── styles.css
```

جزئیات

با شروع به تایپ کردن در `input` یک لیستی از پیشنهادات برای کاربر ظاهر می‌شود با کلیک کردن بر روی هر کدام از آنها، `input` مقدار دهی می‌شود. تغییرات لازم را در فایل `app.js` اعمال کنید.

نکات

- توجه داشته باشید هر پیشنهاد باید در یک عنصر `div` با کلاس `item` ساخته شوند.
- در صورت یافتن نشدن یک پیشنهاد می‌باشد `Not Found!` در یک عنصر با کلاس `not-found` ساخته شود.
- به محض کلیک شدن بر روی یک پیشنهاد مقدار داخل `input` به آن تغییر می‌کند.
- در صورت کلیک شدن بر روی `body` باید لیست ظاهر شده از بین برود.
- توجه کنید که داوری خودکار بر مبنای نام کلاس‌های انجام می‌شود.
- پروژه را با ساختار زیر ارسال کنید.

```
[your-zip-file-name].zip
└── app.js
```