

لغتنامه دو حرفی (الگوریتمی)

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

همه‌ی رشته‌های تولید شده با حروف a و b را اول بر حسب تعداد کاراکترها و سپس به ترتیب لغتنامه‌ای مرتب کردیم. رشته‌های اول به این ترتیب شروع و تا بی‌نهایت ادامه پیدا می‌کنند:

رشته	شماره
a	۱
b	۲
aa	۳
ab	۴
ba	۵
bb	۶
aaa	۷
aab	۸
aba	۹
abb	۱۰
baa	۱۱
bab	۱۲

شماره	رشته
۱۳	bba
۱۴	bbb
۱۵	aaaa
۱۶	aaab
۱۷	...

حال به شما عدد n داده می‌شود و از شما کاراکتر آخر رشته‌ی n ام را پرسیده می‌شود.

ورودی

در یک سطر ورودی، عدد صحیح و مثبت n داده می‌شود.

$$1 \leq n \leq 100$$

خروجی

در یک سطر خروجی، کاراکتر آخر رشته‌ی n ام را چاپ کنید.

مثال‌ها

ورودی نمونه ۱

3

خروجی نمونه ۱

a

رشته‌ی سوم `aaa` و حرف آخر آن `a` است.

ورودی نمونه ۲

16

خروجی نمونه ۲

b

رشته‌ی شانزدهم `aaab` و حرف آخر آن `b` است.

▼ اشتباهات متداول

▼ چک کردن شرایط ورودی مسئله

نیازی نیست چک کنید شرایط گفته شده در ورودی برقرار است یا نه. توضیحات محدودیت‌ها فقط برای آگاهی شما درباره‌ی تست‌ها و محدودیت‌های مسئله است و قطعاً در ورودی‌های داده شده به برنامه‌ی شما رعایت می‌شوند. پس نیازی نیست بنویسید:

```
1 | if 1 <= n <= 100:  
2 |     # answer of problem  
3 | else:  
4 |     # print('invalid input')
```

▼ ابتدا همه‌ی ورودی را گرفتن و در نهایت همه‌ی خروجی را چاپ کردن

شما می‌توانید لابه‌لای دریافت ورودی، خروجی دهید. پس نیازی نیست ابتدا همه‌ی ورودی‌ها را دریافت کنید و در نهایت همه‌ی خروجی‌ها را چاپ کنید. مخصوصاً برای سوالاتی که باید به چندین

سوال پاسخ دهید، می‌توانید دو قسمت ورودی و خروجی را کاملاً مستقل در نظر بگیرید و مطمئن باشید تداخلی پیش نمی‌آید.

▼ چاپ کردن موارد اضافه برای دریافت ورودی

لطفاً از چاپ کردن موارد اضافه مثل `please enter a number` برای دریافت ورودی پرهیز کنید. برای مثال در زبان پایتون نباید بنویسید:

```
1 | input('please enter:')
```

▼ چند فایلی کد زدن

برای زبان‌هایی مثل جاوا نباید در بالای کد شما آدرس پکیج داده شود. برای مثال در بالای کد خود نباید بنویسید:

```
1 | package ir.quera.contest;
```

▼ استفاده از چند `Scanner` برای دریافت ورودی

در زبان جاوا، باید فقط یک شئ از جنس `Scanner` تعریف کنید و همهی ورودی‌ها را با آن دریافت کنید.

▼ نحوه دریافت ورودی و چاپ کردن خروجی

برای آشنایی بیشتر برای نحوه دریافت ورودی و چاپ کردن خروجی این لینک را مطالعه کنید.

بازی اسنوکر (پیاده‌سازی)

- محدودیت زمان: ۱ ثانیه

- محدودیت حافظه: ۲۵۶ مگابایت

در یک بازی اسنوکر، مجموعاً ۱۲ توپ استفاده می‌شود؛ ۱ توپ سفید، ۱۵ توپ قرمز و ۶ توپ رنگی (قرمز و سفید را رنگی در نظر نگیرید ولی مشکی رنگی است). اطلاعات این توپ‌ها به شرح زیر است:

ردیف	توضیح	توپ سفید
۰ امتیاز	white	
۱ امتیاز	red	توپ قرمز
۲ امتیاز	yellow	توپ زرد
۳ امتیاز	green	توپ سبز
۴ امتیاز	brown	توپ قهوه‌ای
۵ امتیاز	blue	توپ آبی
۶ امتیاز	pink	توپ صورتی
۷ امتیاز	black	توپ مشکی

برای کسب امتیاز، ابتدا باید سعی کنید که یک توپ قرمز را پاکت کنید (وارد سوراخ کنید). پس از پاکت شدن توپ قرمز، نوبت پاکت کردن یکی از توپ‌های رنگی به دلخواه است. پس از پاکت توپ رنگی، آن توپ رنگی دوباره به میز باز می‌گردد.

بازی به همین ترتیب ادامه پیدا می‌کند. اگر تمام توپ‌های قرمز پاکت شوند و دیگر توپ قرمزی باقی نماند، می‌توانیم توپ‌های رنگی را وارد کنیم؛ در این صورت توپ‌ها به میز باز نمی‌گردند. توپ سفید همیشه

به میز باز می‌گردد.

اگر در طول بازی، بازیکن توپ سفید را وارد سوراخ کند، یا نتواند توپی را وارد سوراخ کند و یا توپی را خلاف قوانین وارد بازی کند (مثلاً باید قبل از توپ رنگی توپ قرمز وارد کرده باشد) امتیاز توپ وارد شده را نمی‌گیرد و نوبت حریف می‌شود.

بازیکنان هر بار که توپ را با موفقیت پاکت کنند، امتیاز می‌گیرند.

در این سوال برای پایان بازی نیازی نیست همه‌ی توپ‌ها پاکت شده باشند و برنده بازی با جمع امتیازها مشخص می‌شود. دو بازیکن بعد از بازی اسنوکر به سراغ شما می‌آیند و دنباله‌ی نتیجه‌ی ضربه‌ها را به شما می‌گویند و از شما می‌خواهند که نتیجه بازی که یکی از حالت‌های برد بازیکن اول (First)، برد بازیکن دوم (Second) یا تساوی (Tie) است، را مشخص کنید.

توجه کنید ممکن است هیچ راهی برای درست در نظر گرفتن قوانین وجود نداشته باشد، در این حالت Invalid چاپ کنید.

برای بهتر متوجه شدن سوال، به مثال‌ها مراجعه کنید.

ورودی

ابتدا در خط اول یک عدد n داده می‌شود که اندازه‌ی دنباله‌ی ضربه‌ها را مشخص می‌کند.

$$1 \leq n \leq 100$$

سپس در n خط بعدی، در هر خط یک رنگ داده می‌شود و یا کلمه‌ی miss نوشته می‌شود که یعنی توپی در این ضربه پاکت نشده است.

خروجی

در یک خط برنده بازی را مشخص کنید، اگر نفر اول برنده است، First، اگر نفر دوم برنده است، Second و اگر بازی به تساوی رسیده، Tie و در صورتی که ورودی‌ها با قوانین بازی تناقض دارند، Invalid را چاپ

کنید.

مثال‌ها

ورودی نمونه ۱

10
red
black
white
red
blue
green
red
miss
red
yellow

خروجی نمونه ۱

Tie

۱. بازیکن اول توپ قرمز را پاکت می‌کند؛ پس ۰-۱ نتیجه فعلی است.
۲. بازیکن اول توپ سیاه را پاکت می‌کند. چون این توپ رنگی است و توپ قبلی را قرمز وارد کرده، ۷ امتیاز آن را می‌گیرد؛ پس نتیجه بازی ۰-۸ می‌شود. بعد از آن چون هنوز توپ‌های قرمز تمام نشده، توپ مشکی به میز بر می‌گردد.
۳. بازیکن اول توپ سفید را پاکت می‌کند؛ پس هیچ امتیازی نمی‌گیرد و نوبت بازیکن دوم می‌شود.
۴. بازیکن دوم توپ قرمز را پاکت می‌کند؛ پس نتیجه بازی ۱-۸ می‌شود.
۵. بازیکن دوم توپ آبی را پاکت می‌کند؛ پس به دلیل مشابهه ضربه ۲، امتیاز آن را می‌گیرد و نتیجه بازی ۶-۸ می‌شود.

۶. بازیکن دوم توپ سبز را پاکت می‌کند ولی توپ قبلی قرمز نبوده؛ پس توپ سبز را مجدداً به میز بازی بر می‌گردانند و هیچ امتیازی کسب نمی‌کند و نوبت به بازیکن اول بر می‌گردد.
۷. بازیکن اول توپ قرمز را پاکت می‌کند؛ پس نتیجه بازی ۹-۶ می‌شود.
۸. بازیکن اول موفق نمی‌شود در این ضربه توپی را پاکت کند؛ پس نوبت به بازیکن دوم می‌رسد.
۹. بازیکن دوم یک توپ قرمز پاکت می‌کند؛ پس نتیجه بازی به ۹-۷ تغییر می‌کند.
۱۰. بازیکن دوم توپ زرد را پاکت می‌کند؛ پس به دلیل مشابه ضربه ۲، امتیاز آن را می‌گیرد و نتیجه بازی ۹-۹ می‌شود و توپ زرد به میز بر می‌گردد.

در نهایت بازی با نتیجه‌ی تساوی (Tie) به پایان می‌رسد.

ورودی نمونه ۲

10
red
black
red
miss
red
blue
red
red
yellow
red

خروجی نمونه ۲

Second

۱. بازیکن اول توپ قرمز را پاکت می‌کند؛ پس ۱-۰ نتیجه فعلی است.
۲. بازیکن اول توپ سیاه را پاکت می‌کند. چون این توپ رنگی است و توپ قبلی را قرمز وارد کرده، ۷ امتیاز آن را می‌گیرد؛ پس نتیجه بازی ۰-۸ می‌شود. بعد از آن چون هنوز توپ‌های قرمز تمام نشده، توپ مشکی به میز بر می‌گردد.

۳. بازیکن اول توب قرمز را پاکت می‌کند؛ پس ۱ امتیاز می‌گیرد. نتیجه فعلی ۹-۰ و نوبت بازیکن اول می‌ماند.

۴. بازیکن اول هیچ توپی را پاکت نمی‌کند؛ پس نوبت هیچ امتیازی نمی‌گیرد و بازیکن دوم می‌شود.

۵. بازیکن دوم توب قرمز را پاکت می‌کند؛ پس ۱ امتیاز می‌گیرد. نتیجه فعلی ۹-۱ و نوبت بازیکن دوم می‌ماند.

۶. بازیکن دوم توب آبی را پاکت می‌کند و چون توب قبلی را قرمز وارد کرده؛ پس ۵ امتیاز آن را می‌گیرد. نتیجه بازی ۹-۶ می‌شود و بعد از آن چون هنوز توپ‌های قرمز تمام نشده، توب آبی به میز بر می‌گردد.

۷. بازیکن دوم توب قرمز را پاکت می‌کند؛ پس نتیجه بازی ۹-۷ می‌شود.

۸. بازیکن دوم توب قرمز را پاکت می‌کند؛ پس نتیجه بازی ۹-۸ می‌شود.

۹. بازیکن دوم یک توب زرد پاکت می‌کند؛ پس ۲ امتیاز می‌گیرد و نتیجه بازی به ۱۰-۹ تغییر می‌کند.

۱۰. بازیکن دوم توب قرمز را پاکت می‌کند؛ پس نتیجه بازی ۹-۱۱ می‌شود.

در نهایت بازی با نتیجه‌ی برد بازیکن دوم (Second) به پایان می‌رسد.

ورودی نمونه ۳

1
red

خروجی نمونه ۳

First

بازیکن اول توب قرمز را پاکت می‌کند. بازی ۱-۰ و با نتیجه‌ی برد بازیکن اول (First) به پایان می‌رسد.

ورودی نمونه ۴

17
red

red
red
red
red
red
red
red
red
red
red
red
red
red
red
yellow
yellow

خروجی نمونه ۱۴

Invalid

در این حالت بازیکن اول همه‌ی ۱۵ توپ قرمز را پاکت می‌کند. در نتیجه اگر توپ رنگی وارد شود دیگر به میز بر نمی‌گردد ولی بعد از آن دو بار توپ زرد پاکت شده و این طبق قوانین ممکن نیست. بنابراین پاسخ Invalid است.

تیم همیشه حاضر (الگوریتمی)

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

برنامهنویس‌های شرکت یکتانت به تعدادی تیم تقسیم شده‌اند و در یک صف کنار هم ایستاده‌اند. یک تیم، «همیشه حاضر» است، اگر و تنها اگر در بین هر k نفر متوالی از افراد داخل صف، حداقل یکی از افراد این تیم در بین این افراد باشد. کمترین مقدار k را بیابید که حداقل یک تیم «همیشه حاضر» داشته باشیم.

ورودی

در سطر اول ورودی، عدد صحیح n داده می‌شود که نشان‌دهنده‌ی تعداد برنامهنویس‌های شرکت یکتانت است.

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

در سطر دوم ورودی، شماره‌ی تیم‌های این صف به ترتیب داده می‌شود که همگی اعداد طبیعی کمتر یا مساوی ۱۰۰ ۰۰۰ است.

خروجی

کمترین مقدار k را بیابید که حداقل یک گروه همیشه حاضر داشته باشیم.

مثال‌ها

ورودی نمونه ۱

1 2 3 1 3 1

خروجی نمونه ۱

3

در هر سه نفر متوالی، حداقل یک نفر از تیم ۱ وجود دارد. همچنین هیچ تیمی نیست که برای هر دو نفر متوالی در صفر، یک نفر از آنها آمده باشد. بنابراین کمترین k ممکن برابر ۳ است.

ورودی نمونه ۲

3

1 2 3

خروجی نمونه ۲

2

از هر دو نفر متوالی، حداقل یک نفر از تیم ۲ وجود دارد. چنین خاصیتی برای هر نفر وجود ندارد. بنابراین کمترین k ممکن برابر ۲ است.

پیشنهاد جایگاه (پیاده‌سازی)

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

فرض کنید سامانه‌ای داریم که شامل تعدادی تبلیغ و وبسایت است که این وبسایتها را به عنوان جایگاهی برای این تبلیغات در نظر می‌گیریم. هرکدام از این جایگاهها و تبلیغات دارای تعدادی برچسب مختلف مثل «Football»، «Technology» و... هستند. هدف اصلی این سامانه این است که این تبلیغات و جایگاهها را مدیریت کرده و هر تبلیغ را به جایگاه مناسبی متصل کند.

نحوه پرداخت هزینه‌ی تبلیغات به صورت «هزینه به ازای هر کلیک» (*Cost Per Click*) است که به اختصار *CPC* نامیده می‌شود. هر جایگاه و هر تبلیغ یک *CPC* مورد انتظار دارند که در ادامه سوال با آن آشنا می‌شوید.

شما باید برنامه‌ای برای مدیریت این سامانه بنویسید که با گرفتن لیستی از درخواست‌ها، آنها را به ترتیب اجرا کرده و خروجی مربوط به آن را چاپ کند. جزئیات مربوط به این درخواست‌ها به تفکیک نوع آنها در ادامه آمده است.

برچسب

▼ اضافه کردن برچسب

```
ADD-TAG -name <name>
```

این درخواست اضافه کردن یک برچسب با نام `<name>` است.

- `<name>`: رشته‌ای با حروف الفبای انگلیسی بزرگ و کوچک و اعداد و بدون فاصله با طول حداقل ۳۰ کاراکتر.

به هر برچسب بعد از اضافه شدن یک شماره به نام `id` نسبت داده می‌شود، این شماره‌ها از ۱ شروع می‌شوند و به ترتیب افزایش می‌یابند.

- اگر نام برچسب تکراری باشد باید Error: Tag already exists را چاپ کنید.
- اگر خطای رخ نداد و شماره‌ی این برچسب Done: Tag id is <id> بود باید <id> را چاپ کنید.

▼ لیست تمام برچسب‌ها

TAG-LIST

این درخواست نمایش لیست همه‌ی برچسب‌ها است. خروجی باید ابتدا کلمه‌ی TAGs: و سپس نام برچسب‌ها به ترتیب شماره‌ی آن‌ها با فاصله از هم باشد. مثل:

TAGs: <tag1> <tag2> ...

تبلیغ

▼ اضافه شدن یک تبلیغ

<ADD-ADS -name <name> -cpc <cpc> -tags <tag1> <tag2> ...

این درخواست اضافه کردن یک تبلیغ با نام <name>، با مقدار CPC مورد انتظار <cpc>، لیست برچسب‌های مربوط به این تبلیغ <tag1>، <tag2> و... است.

- <name> : رشته‌ای با حروف الفبای انگلیسی بزرگ و کوچک و اعداد و بدون فاصله با طول حداقل ۳۰ کارکتر.
- <cpc> : یک عدد صحیح نامنفی حداقل ۱۰۰۰ است.
- <tag1> <tag2> ... : لیستی از نام موضوعات.

به هر تبلیغ بعد از اضافه شدن یک شماره به نام id نسبت داده می‌شود، این شماره‌ها از ۱ شروع می‌شوند و به ترتیب افزایش می‌یابند.

- Error: Ad already exists: اگر تبلیغ تکراری باشد:

- اگر موضوعی وجود نداشته باشد: Error: Tag not found
- اگر خطای رخ نداد و شماره‌ی این تبلیغ <id> بود باید Done: Ads id is <id> را چاپ کنید.

▼ لیست تمامی تبلیغات

ADS-LIST

این درخواست نمایش لیست همه‌ی تبلیغات است. خروجی باید ابتدا کلمه‌ی ADSS: و سپس نام تبلیغات به ترتیب شماره‌ی آن‌ها با فاصله از هم باشد. مثل:

ADSS: <ads1> <ads2> ...

جایگاه

▼ اضافه شدن یک جایگاه

ADD-PLACE -name <name> -cpc <cpc> -tags <tag1> <tag2> ...

این درخواست اضافه کردن یک جایگاه با نام <name>، با مقدار CPC مورد انتظار <cpc>، لیست برچسب‌های مربوط به این جایگاه <tag1>، <tag2> و... است.

- <name> : رشته‌ای با حروف الفبای انگلیسی بزرگ و کوچک و اعداد و بدون فاصله با طول حداقل ۳۰ کarakتر.
- <cpc> : یک عدد صحیح نامنفی حداقل ۱۰۰۰ است.
- <tag1> <tag2> ... : لیستی از نام موضوعات.

به هر جایگاه بعد از اضافه شدن یک شماره به نام id نسبت داده می‌شود، این شماره‌ها از ۱ شروع می‌شوند و به ترتیب افزایش می‌یابند.

- اگر جایگاه تکراری باشد: Error: Place already exists
- اگر موضوعی وجود نداشته باشد: Error: Tag not found

- اگر خطای رخ نداد و شماره‌ی این سایت `Place id is <id>` بود باید `<id>` را چاپ کنید.

▼ لیست تمامی جایگاهها

PLACE-LIST

این درخواست نمایش لیست همه‌ی جایگاهها است. خروجی باید ابتدا کلمه‌ی `PLACES:` و سپس نام تبلیغات به ترتیب شماره‌ی آن‌ها با فاصله از هم باشد. مثل:

`PLACES: <place1> <place2> ...`

پیشنهادها

▼ پیشنهاد تبلیغ به جایگاه

SUGGEST-ADS -id <id>

این درخواست لیست تمام تبلیغات به ترتیب مناسب بودن برای جایگاه `<id>` است.

- اگر جایگاه وجود نداشت: `Error: Place not found`
- در غیر این صورت ابتدا عبارت `SUGGEST-ADS:` و سپس شماره‌ی همه‌ی تبلیغات را در یک سطر و با فاصله از هم به ترتیب مناسب بودن چاپ کنید.

مناسب بودن تبلیغ شماره‌ی i برای جایگاه شماره‌ی j از فرمول زیر محاسبه می‌شود و هر چه این عدد بیشتر باشد، تبلیغ مناسب‌تری است.

$$\frac{1}{\max\{1, cpc_i - cpc_j\}} \times (\text{Number of Matched Tags} - \text{Number of Unmatched Tags})$$

- منظور از cpc_i مقدار مورد انتظار CPC برای تبلیغ شماره‌ی i است.
- منظور از cpc_j مقدار مورد انتظار CPC برای جایگاه شماره‌ی j است.

- منظور از Number of Matched Tags تعداد برچسب‌های مشترک بین تبلیغ شماره‌ی i و جایگاه شماره‌ی j است.
- منظور از Number of Unmatched Tags تعداد برچسب‌هایی از تبلیغ شماره i است که در لیست برچسب‌های جایگاه شماره j نیامده است.
- اگر دو تبلیغ امتیاز یکسانی داشتند، آن‌ها را به ترتیب شماره‌هایشان مرتب کنید.

▼ پیشنهاد جایگاه به تبلیغ

SUGGEST-PLACE -id <id>

این درخواست لیست تمام جایگاه‌ها به ترتیب مناسب بودن برای تبلیغ <id> است.

- اگر تبلیغ وجود نداشت: Error: Ads not found
- در غیر این صورت ابتدا عبارت SUGGEST-PLACE: و سپس شماره‌ی همه‌ی جایگاه‌ها را در یک سطر و با فاصله از هم به ترتیب مناسب بودن چاپ کنید.

مناسب بودن جایگاه شماره‌ی i برای تبلیغ شماره‌ی j از فرمول زیر محاسبه می‌شود و هر چه این عدد بیشتر باشد، تبلیغ مناسب‌تری است.

$$\frac{1}{\max\{1, cpc_i - cpc_j\}} \times (\text{Number of Matched Tags} - \text{Number of Unmatched Tags})$$

- منظور از cpc_i مقدار مورد انتظار CPC برای جایگاه شماره‌ی i است.
- منظور از cpc_j مقدار مورد انتظار CPC برای تبلیغ شماره‌ی j است.
- منظور از Number of Matched Tags تعداد برچسب‌های مشترک بین جایگاه شماره‌ی i و تبلیغ شماره‌ی j است.
- منظور از Number of Unmatched Tags تعداد برچسب‌هایی از جایگاه شماره‌ی i است که در لیست برچسب‌های تبلیغ j نیامده است.
- اگر دو جایگاه امتیاز یکسانی داشتند، آن‌ها را به ترتیب شماره مرتب کنید.

▼ متصل کردن تبلیغ به جایگاه

```
MATCH -ads-id <ads-id> -place-id <place-id>
```

- اگر تبلیغ وجود نداشت: Error: Ads not found
- اگر جایگاه وجود نداشت: Error: Place not found
- در غیر این صورت: Done: <ads-id> matched to <place-id>

توجه کنید بعد از متصل کردن، باید این تبلیغ و جایگاه را از لیست تبلیغات و جایگاه‌های موجود حذف کنید.

ورودی

در سطر اول ورودی، عدد صحیح n که تعداد درخواست‌ها را نشان می‌دهد آمده است.

$$1 \leq n \leq 100$$

در n سطر بعدی، در هر سطر یکی از درخواست‌ها داده می‌شود.

خروجی

خروجی هر درخواست را به ترتیب چاپ کنید.

مثال

ورودی نمونه ۱

21

```
ADD-TAG -name Football
ADD-TAG -name Technology
ADD-TAG -name Sports
ADD-TAG -name Football
```

```
TAG-LIST
ADD-ADS -name Tv -cpc 500 -tags Football
ADD-ADS -name Ps -cpc 700 -tags Technology Football
ADD-ADS -name Tv -cpc 600 -tags Technology
ADS-LIST
ADD-PLACE -name Ineternet -cpc 600 -tags Football
ADD-PLACE -name Street -cpc 500 -tags Technology
ADD-PLACE -name School -cpc 800 -tags Sports
ADD-PLACE -name Ineternet -cpc 700 -tags Sports
PLACE-LIST
SUGGEST-ADS -id 3
SUGGEST-ADS -id 2
SUGGEST-PLACE -id 2
MATCH -ads-id 1 -place-id 1
MATCH -ads-id 3 -place-id 2
MATCH -ads-id 2 -place-id 4
MATCH -ads-id 2 -place-id 2
```

خروجی نمونه ۱

```
Done: Tag id is 1
Done: Tag id is 2
Done: Tag id is 3
Error: Tag already exists
TAGs: Football Technology Sports
Done: Ads id is 1
Done: Ads id is 2
Error: Ad already exists
ADSs: Tv Ps
Done: Place id is 1
Done: Place id is 2
Done: Place id is 3
Error: Place already exists
PLACEs: Ineternet Street School
SUGGEST-ADS: 1 2
SUGGEST-ADS: 2 1
SUGGEST-PLACE: 1 2 3
Done: 1 matched to 1
```

Error: Ads not found

Error: Place not found

Done: 2 matched to 2

- چهار برچسب اضافه می‌شود و تلاش برای اضافه کردن برچسب تکراری Football خطأ می‌دهد.
- لیست برچسب‌ها چاپ می‌شود.
- دو تبلیغ اضافه می‌شود و تلاش برای اضافه کردن تبلیغ تکراری TV خطأ می‌دهد.
- لیست تبلیغات چاپ می‌شود.
- سه جایگاه اضافه می‌شود و تلاش برای اضافه کردن جایگاه تکراری Internet خطأ می‌دهد.
- لیست جایگاه‌ها چاپ می‌شود.
- درخواست پیشنهاد تبلیغ برای جایگاهی که وجود ندارد خطأ می‌دهد.
- پیشنهادات تبلیغ برای جایگاه ۱ و پیشنهادات جایگاه برای تبلیغ ۲ نمایش داده می‌شود.
- تبلیغ ۱ به جایگاه ۱ متصل می‌شود.
- تلاش برای اتصال تبلیغی که وجود ندارد خطأ می‌دهد.
- تلاش برای اتصال به جایگاهی که وجود ندارد خطأ می‌دهد.
- تبلیغ ۲ به جایگاه ۲ متصل می‌شود.

کیک تکه تکه (الگوریتمی)

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

بیژن یک کیک مستطیلی برای تولد کوئرا خریداری کرده است و کیک را با تعدادی برش افقی و عمودی موازی اصلاح مستطیل، به تکه‌های مستطیلی تقسیم کرده است.

منیژه با تعدادی از دوستان خود به جشن آمده است. او برای دوستان خود بزرگترین تکه‌ها را جدا کرده و به هر نفر یک تکه کیک می‌دهد. سپس برای خودش بزرگترین تکه کیک باقی مانده را برمی‌دارد.

می‌دانیم عرض کیک با برش‌های افقی، و طول کیک با برش‌های عمودی به تعدادی قسمت تقسیم شده است. تعدادی سناریو داریم. در هر سناریو، منیژه در ابتدا تعداد افراد گروه خود که شامل دوستانش و خودش است را به شما می‌گوید. سپس تعداد قسمت‌هایی که عرض و طول کیک تقسیم شده‌اند و طول هر قسمت را می‌گوید و از شما می‌خواهد به او بگویید مساحت تکه کیک او چقدر خواهد بود.

ورودی

در ابتدا در خط اول عدد q ، که نشانگر تعداد سناریوهای است داده می‌شود.

$$1 \leq q \leq 100\,000$$

هر سناریو در ۳ خط ورودی داده می‌شود. در ابتدا در خط اول هر سناریو، اعداد n و m و k ورودی داده می‌شوند که به ترتیب تعداد قسمت‌های تقسیم شده عرض و طول کیک و عدد k نیز جمعیت گروه منیژه و دوستانش را مشخص می‌کنند.

$$1 \leq n, m \leq 100\,000, \quad 1 \leq k \leq mn$$

سپس در خط دوم به ترتیب n عدد، که طول قسمت‌هایی که عرض کیک برش داده شده است را مشخص می‌کند ورودی داده می‌شوند.

$$1 \leq h_i \leq 10^9$$

و در خط سوم m عدد، که طول مستطیل کیک برش داده شده است، داده می‌شود.

$$1 \leq v_i \leq 10^9$$

تضمین می‌شود که مجموع $n + m$ روی همهٔ سناریوها حداقل ۱۰۰,۰۰۰ باشد.

خروجی

در q خط، و در هر خط یک عدد که نشانگر مساحت k امین بزرگترین تکه کیک آن سناریو است را خروجی دهیم.

مثال‌ها

ورودی نمونه ۱

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

خروجی نمونه ۱

12

60

14

فوتبالیستا (پایگاهداده)

کد شما باید روی PostgreSQL قابل اجرا باشد.

در این سوال، بخشی از پایگاه داده مربوط به مسابقات فوتبال اروپا در اختیار شما قرار گرفته است.

جزئیات پایگاهداده

داده‌های اولیه برای تست نهایی را از این لینک [دانلود کنید](#).

▼ توضیحات در مورد داده‌های اولیه

در فایل `football.zip` فایلی به اسم `initial.sql` وجود دارد.

ابتدا پایگاهداده‌ای با نام `football` در سیستم خود را بسازید و با اجرای این فایل برروی این پایگاهداده، همه جداول و سطرهایی که برای تست نهایی مورد استفاده قرار می‌گیرد در سیستم شما ایجاد می‌شود.

▼ توضیحات جداول دیتابیس

ساختار جداول به شرح زیر است:

جدول `players`: از این جدول برای نگهداری اطلاعات بازیکن‌ها استفاده می‌شود. ساختار این جدول به صورت زیر است:

نام ستون	نوع	تعریف
<code>player_id</code>	<code>integer</code>	شناسه‌ی بازیکن
<code>current_club_id</code>	<code>integer</code>	شناسه‌ی پیشگاه فعلی بازیکن
<code>player_code</code>	<code>character varying(64)</code>	کد بازیکن (نمایه بازیکن)
<code>country_of_birth</code>	<code>character varying(32)</code>	کشور تولد بازیکن

نام ستون	نوع	تعریف
city_of_birth	character varying(64)	شهر تولد بازیکن
country_of_citizenship	character varying(32)	کشور ملیت بازیکن
date_of_birth	date	تاریخ تولد بازیکن
sub_position	character varying(32)	شخص دوم بازیکن
position	character varying(16)	شخص اول بازیکن
foot	character varying(8)	پای شخصی بازیکن
height_in_cm	integer	قد بازی کن به سانتی متر
contract_expiration_date	date	تاریخ انقضای قرارداد بازیکن

جدول clubs : از این جدول برای نگهداری اطلاعات باشگاهها استفاده می‌شود. ساختار این جدول به صورت زیر است:

نام ستون	نوع	تعریف
club_id	integer	شناسه‌ی باشگاه
name	character varying(64)	نام باشگاه
domestic_competition_id	character varying(4)	ریک پاشگاه
squad_size	integer	اندازه تیم
foreigners_number	integer	تعداد افراد خارجی تیم
national_team_players	integer	تعداد بازیکن‌های تیم ملی
stadium_name	character varying(64)	نام استادیوم اختصاصی باشگاه

نام ستون	نوع	تعریف
stadium_seats	integer	کشور ملیت بازیکن
net_transfer_record	character varying(10)	ارزش خارج پاشکنای

جدول competitions : از این جدول برای نگهداری اطلاعات مسابقات استفاده می‌شود. ساختار این جدول به صورت زیر است:

نام ستون	نوع	تعریف
competition_id	character varying(4)	شناسه مسماط
name	character varying(64)	نام مسماط
type	character varying(32)	دسته مسماط
country_name	character varying(10)	کشور مسماط

جدول games : از این جدول برای نگهداری اطلاعات بازی‌ها استفاده می‌شود. ساختار این جدول به صورت زیر است:

نام ستون	نوع	تعریف
game_id	integer	شناسه بازی
competition_id	character varying(4)	شناسه تیم پاشکنای
season	integer	فصل برگزاری بازی
date	date	تاریخ بازی
home_club_id	integer	شناسه تیم (پاشکنای) میزبان
away_club_id	integer	شناسه تیم مهمان

نام ستون	نوع	تعریف
home_club_goals	integer	تعداد گل تیم صیغه ای
away_club_goals	integer	تعداد گل تیم مددگار
stadium	character varying<64>	نام استادیوم بازی
attendance	integer	تعداد تماشگرها

جدول **appearances** : از این جدول برای نگهداری اطلاعات حضور بازیکن‌ها در بازی استفاده می‌شود.
ساختار این جدول به صورت زیر است:

نام ستون	نوع	تعریف
appearance_id	character varying(16)	شناسه‌ی جدول
game_id	integer	شناسه‌ی بازی
player_id	integer	شناسه‌ی بازیکن
yellow_cards	integer	تعداد کارت زردی‌ای بازیکن
red_cards	integer	تعداد کارت سرمه‌نهای بازیکن
goals	integer	تعداد گلهای بازیکن
assists	integer	تعداد پاس گل‌های بازیکن
minutes_played	integer	تعداد دقیقه که بازیکن بازی کرده است

جدول **game_events** : از این جدول برای نگهداری اطلاعات اتفاق‌های مسابقه (گل، موقعیت گل، پنالتی و ...) استفاده می‌شود. ساختار این جدول به صورت زیر است:

نام ستون	نوع	تعریف
game_event_id	integer	شناسه‌ی جدول
game_id	integer	شناسه‌ی بازی
minute	integer	زمان(دقیقه) اتفاق
type	character varying(16)	نوع اتفاق
player_id	integer	شناسه‌ی پاریکن اصلی اتفاق
player_in_id	integer	شناسه‌ی پاریکن که از کل جمیع پیروی نموده است
player_assist_id	integer	شناسه‌ی پاریکن که پاس کل داده است

مطلوبات

۱. لیستی از ارزش خالص باشگاه را به صورت عددی (integer) .

▼ توضیحات مربوط به کوئری اول

نام باشگاه را در ستونی با نام `name` و ارزش خالص هر باشگاه را در ستونی با نام `total` نمایش دهید. توجه کنید خروجی شما باید به ترتیب نزولی بر حسب ارزش خالص باشگاه و در صورت که این مقدار برابر بود بر اساس نام باشگاه به صورت صعودی مرتب شود.

▼ توجه

دقت داشته باشید در صورتی که ارزش خالص تیم مقدار `-0` داشت آن را به صورت `0` نمایش دهید. همچنین نماد `\k` معادل هزار و نماد `\m` معادل میلیون می باشد.

3 سطر اول خروجی شما باید به شکل زیر باشد.

name	total
Villarreal Club de Fútbol S A D	99400000
Brighton and Hove Albion Football Club	86400000
Verona Hellas Football Club	84300000

۲. مجموع تعداد کارت‌های زرد و قرمز هر بازیکن در هر سری مسابقات را نمایش دهید.

▼ توضیحات مربوط به کوئری دوم

نام یا همان کد بازیکن را در ستونی به نام `player_name` و نام سری مسابقات را در ستونی با نام `competition_name`، تعداد کارت زردهای بازیکن در آن سری مسابقات را در ستونی با نام `total_yellow_cards` و تعداد کارت زردهای بازیکن در آن سری مسابقات را در ستونی با نام `total_red_cards` قرار دهید. ستون‌ها را ابتدا بر حسب نام بازیکن به صورت **صعودی**، سپس بر حسب تعداد کارت‌های زرد و سپس قرمز، به صورت **نزولی** و در نهایت براساس نام سری مسابقات به صورت **صعودی** مرتب کنید.

▼ توجه

هنگامی که مجموع کارت‌های زرد و مجموع کارت‌های زرد قرمز در سری مسابقاتی برای بازیکنی صفر باشد (بازیکن هیچ کارتی دریافت نکرده باشد)، را نمایش ندهید.

3 سطر اول خروجی شما باید به شکل زیر باشد.

player_name	competition_name	total_yellow_cards	total_red_cards
aaron_appindangoye	super_lig	0	0
aaron_appindangoye	europea_league	1	0
aaron_boupendza	super_lig	7	0

۳. سری مسابقات را بر حسب مجموع تعداد اتفاقهای آن (تعداد گل‌ها، موقعیت‌های گل، کارت‌ها و ...) رتبه بندی کنید.

▼ توضیحات مربوط به کوئری سوم

رتبه سری مسابقات بر اساس مجموع تعداد اتفاقهای آن (پر حادثه‌ترین رتبه اول) را در ستونی با نام ranking و نام سری مسابقات را در ستونی با نام name و تعداد اتفاقهای در آن سری مسابقات را در ستونی با نام events قرار دهید. ستون‌ها را ابتدا بر حسب رتبه، سپس براساس نام سری مسابقات به صورت صعودی مرتب کنید.

▼ توجه

اگر رتبه دو سری مسابقات با بیشترین تعداد حوادث، یکسان بود، هردو رتبه یک می‌شوند و سری مسابقات بعدی رتبه‌اش دو می‌شود، این قانون برای تمامی رتبه‌ها صادق است.

3 سطر آخر خروجی شما باید به شکل زیر باشد.

ranking	name	events
39	johan_cruyff_school	47
40	trophee_des_champions	45
41	ukrainian_super_cup	33

۴. لیستی از بازی‌ها، استادیوم آن بازی و بازیکن‌هایی که هنگام مسابقه حداقل 35 سال داشته‌اند به همراه تعداد گل‌های زده شده توسط آن بازیکن و سن وی در آن بازی.

▼ توضیحات مربوط به کوئری چهارم

نام یا همان کد بازیکن را در ستونی به نام game_id، شناسه بازی را در ستونی با نام player_code، استادیومی که بازی در آن انجام شده را در ستونی به نام stadium، سن بازیکن را در آن مسابقه را در ستونی با نام age_at و تعداد گل‌های زده شده توسط بازیکن در آن بازی را در ستونی به نام

`total_goals` قرار دهید. ستون‌ها را ابتدا برحسب نام بازیکن بهصورت صعودی ، سپس برحسب سن بازیکن در آن بازی بهصورت نزولی و سپس برحسب تعداد گل بهصورت نزولی، سپس براساس نام سری استادیوم و درنهایت براساس شناسه بازی بهصورت صعودی مرتب کنید.

▼ توجه

فقط بازیکن‌هایی را که حداقل یک گل در این بازی‌ها زده‌اند، نمایش دهید.

<code>player_code</code>	<code>game_id</code>	<code>stadium</code>	<code>age_at</code>	<code>total_goals</code>
adam_le_fondre	4121036	Easter Road Stadium	37	1
adam_le_fondre	4120961	Global Energy Stadium	37	1
adam_le_fondre	4120853	Easter Road Stadium	36	1

روش پیاده‌سازی

در یک فایل با نام `code.sql` کد خود را قرار دهید و آن را فشرده (`zip`) کنید و در سایت بارگذاری نمایید. کد شما باید به صورت زیر باشد(نام فایل `zip` مهم نیست).

```

1  -- Section1
2    Your first query here
3  -- Section2
4    Your second query here
5  -- Section3
6    Your third query here
7  -- Section4
8    Your fourth query here

```