KMAP

- محدودیت زمان: 5 ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

مهدیز که مدتی تنها و گوشه‌گیر شده است و متوجه شده که دوست گمشده‌اش در درس مدار منطقی با چالش روبرو شده و فهمیده که اگر برای او یک ساده کننده عبارت‌های بولی با استفاده از جدول کارنو بنویسد، دوست گمشده‌اش خیلی خوشحال خواهد شد و او را از تنهایی و عزلت در می‌آورد. به مهدیز کمک کنید این کار را انجام دهد قبل از اینکه دوست گمشده‌اش، او را برای همیشه رها کند.

ابتدا تعداد متغیرهای عبارت منطقی (از A شروع و با Z تمام می‌شوند)، و سپس در یک خط شماره مینترم‌های عبارت داده می‌شود. شما باید برنامه‌ای بنویسید که با استفاده از جدول کارنو عبارت را به ساده ترین فرم SOP ساده کنید.

ترتیب termها مهم نیست اما در هر term متغیرها را به ترتیب ASCII چاپ کنید.

ورودی نمونه 1

```
3
3,5,6,7
```

خروجی نمونه 1

```
BC + AC + AB
```

ورودی نمونه 2

```
3
0,2,3
```

خروجی نمونه 2

```
A'C' + A'B
```

Blockchain

- محدودیت زمان: ۴ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

مهدیز که یک دانشجوی ترم ۲ای است، از بی‌پولی در عذاب فراوان به سر می‌بره. وی به دلیل فشار درس و دانشگاه، زمان لازم برای کار کردن و کسب درآمد را نداره و تدریس به بچه دبیرستانی‌ها دیگه کفاف خرجش رو نمیده، از این رو تصمیم گرفته تا از مهارت برنامه‌نویسی خودش استفاده کنه و یک دایره‌ی مالی طراحی کنه که بتونه مشتری‌های زیادی رو جذب کنه؛ اما از طرفی خودش بیشترین سود را ببره!

او در حال برنامه ریزی برای دایره مالی‌اش بود که فهمید باید خودش را برای امتحان شفاهی یکی از دروسش آماده کند؛ از این رو قوانینی که برای دایره مالی تهیه کرده بود را در اختیار شما قرار داد تا در این مهم او را یاری کرده و دست او رو بگیرید.

دایره دارای دستورات و قوانین زیر می‌باشد:

- دایره برای شروع کار نیاز به یک بنیان‌گذار داره که باید مبلغ ۵۰۰۰ دلار برای ایجاد میز به مهدیز پرداخت کنه. اگر بودجه‌ی اون بیشتر از این مبلغ باشه، به عنوان ذخیره در حسابش قرار می‌گیره؛ اما اگر کمتر باشه، نمی‌تونه میز ایجاد کنه.
- به دلیل نظام سرمایه‌داری، دایره‌ی مالی شامل طبقات مختلف هستش. هر طبقه یک میز گرد است که در آن، هر فرد جدید، سمت راست آخرین فرد اضافه شده به میز می‌نشیند.
- بنیان‌گذار در طبقه ۰ قرار داره و بعد از آن، طبقات از ۱ شماره‌گذاری شده‌اند. هر طبقه ظرفیت محدودی از افراد را در خود جای می‌دهد که طبق تابع درجه دو x^2 محاسبه می‌شود و افراد با توجه به طبقه‌ای که در آن قرار دارند، سود متفاوتی دریافت می‌کنند.
- بعد از ایجاد میز توسط بنیان‌گذار سایر افراد از دو طریق می‌توانند وارد دایره شوند:

۱- از طریق معرفی شدن توسط کسانی که از قبل در میز بوده؛ در این شرایط آن‌ها در اولین طبقه دارای ظرفیت پایین‌تر از معرف خود قرار می‌گیرند. مبلغی که با خود به همراه می‌آورند، به این صورت تقسیم می‌شود:

- ۲۰ درصد حساب خودشان

- ۱۰ درصد حساب بنیان‌گذار میز
- ۵ درصد معرف
- ۱۵ درصد به عنوان کارمزد به مهدیز پرداخت می‌شود

و در نهایت ۵۰ درصد باقی‌مانده به صورت مساوی بین طبقات بالایی تقسیم می‌شود. (یعنی اگر ۵ طبقه داشته باشیم، ۲۰ درصد از آن به هر طبقه می‌رسد) و پولی که به هر طبقه رسیده بین اعضای آن به طور مساوی تقسیم می‌شود.

۲- به صورت مستقل که در این صورت به یک طبقه‌ی جدید در انتهای طبقات اضافه می‌شوند. مبلغی که با خود به همراه می‌آورند به این صورت تقسیم می‌شود:

- ۱۵ درصد حساب خودشان
- ۱۰ درصد حساب بنیان‌گذار میز
- ۲۵ درصد به صورت کارمزد به مهدیز پرداخت می‌شود

و در نهایت مقداری که باقی‌مانده به صورت مساوی بین طبقات بالایی تقسیم می‌شود (یعنی اگر ۵ طبقه داشته باشیم، ۲۰ درصد از آن به هر طبقه می‌رسد) و پولی که به هر طبقه رسیده، بین اعضای آن به طور مساوی تقسیم می‌شود.

- اگر فردی ۵ نفر را معرفی کرده و وارد دایره کند، به یک طبقه بالا منتقل می‌شود. اگر طبقه‌ی بالا پر باشد، جای این فرد با فردی از طبقه‌ی بالا که کم‌ترین تعداد معرفی و سپس میزان پول را دارد، عوض می‌شود. پس از هر تغییر طبقه، تعداد افراد معرفی شده فرد، ۰ می‌شود. تضمین می‌شود با بالا رفتن یک نفر، هیچ طبقه‌ای خالی نخواهد شد.
- هر فرد نام کاربری‌ای دارد که با آن شناخته می‌شود.
- در هر لحظه، برنامه باید بتواند هر یک از اطلاعات زیر را در اختیارمان قرار دهد:

۱- تعداد طبقات

۲- تعداد کل اعضا

۳- تعداد اعضای هر طبقه

۴- معرف هر فرد

۵- نفرات سمت چپ و راست هر فرد

۶- موجودی حساب هر فرد

۷- افرادی که با یک فرد سر یک میز هستند

۸- سود مهدیز تا آن لحظه

نکته: مهدیز بنیان‌گذار میز نیست، بلکه سود او از کارمزدهایی که دریافت می‌کند، تامین می‌شود.

ورودی و خروجی

ایجاد یک دایره:

```
Create_a_table_for <username> with_deposit_of <money>
```

کار با این دستور شروع می‌شود.

اگر از قبل بنیان‌گذار داشته باشیم:

```
We already have a founder
```

اگر بودجه فرد کمتر از ۵۰۰۰ دلار باشد:

```
Money is not enough
```

اگر دایره با موفقیت ایجاد شود:

```
You now own a table
```

معرفی یک عضو جدید:

```
Invitation_request_from <username> for <username> with_deposit_of <money>
```

اگر نام کاربری فرد قبلا استفاده شده باشد:

```
Username already taken
```

اگر فرد با موفقیت اضافه شود:

```
User added successfully in level <level>
```

اضافه شدن به صورت مستقل:

```
Join_request_for <username> with_deposit_of <money>
```

اگر نام کاربری فرد قبلا استفاده شده باشد:

```
Username already taken
```

اگر فرد با موفقیت اضافه شود:

```
User added successfully in level <level>
```

تعداد طبقات:

```
Number_of_levels
```

خروجی، تعداد طبقات خواهد بود.

تعداد کل اعضا:

```
Number_of_users
```

خروجی، تعداد اعضا خواهد بود.

تعداد اعضای یک طبقه:

```
Number_of_users_in_level <level>
```

اگر همچین طبقه‌ای موجود نباشد:

```
No_such_level_found
```

در غیر این صورت، خروجی تعداد اعضا در طبقه‌ی داده شده، خواهد بود.

معرف یک فرد:

```
Introducer_of <username>
```

اگر این نام کاربری موجود نباشد:

```
No_such_user_found
```

اگر فرد معرف نداشته باشد:

```
No_introducer
```

در غیر این صورت خروجی نام معرف فرد خواهد بود.

دوستان یک فرد (الکی مثلا کسی که سمت چپ و راست نشسته دوست اون فرده):

```
Friends_of <username>
```

اگر این نام کاربری موجود نباشد:

```
No_such_user_found
```

اگر فرد، تنها و بدون هیچ دوستی دور میز نشسته بود:

```
No_friend
```

در غیر این صورت اگر فرد تنها یک دوست داشت، نام کاربری آن دوست را چاپ کنید و اگر ۲ دوست داشت به ترتیب نام کاربری دوست سمت چپ و راست فرد را در یک خط چاپ کنید و با اسپیس جدا کنید.

موجودی حساب یک فرد:

```
Credit_of <username>
```

اگر این نام کاربری موجود نباشد:

```
No_such_user_found
```

در غیر این صورت خروجی، موجودی حساب فرد خواهد بود.

افرادی که با یک فرد سر یک میز هستند:

```
Users_on_the_same_level_with <username>
```

اگر این نام کاربری موجود نباشد:

```
No_such_user_found
```

اگر فرد در میز تنها بود:

```
He_is_all_by_himself
```

در غیر این صورت خروجی لیستی از نام اعضا که با فرد سر یک میز هستند به جز خود فرد، به ترتیبی که به میز اضافه شده‌اند، خواهد بود که با اسپیس جدا شده‌اند.

سود مهدیز تا آن لحظه:

```
How_much_have_we_made_yet
```

خروجی این دستور یک عدد خواهد بود که برابر با مجموع همه وجوهی‌ست که تا آن لحظه به عنوان کامزد پرداخت شده‌اند.

پایان برنامه:

```
End
```

مثال

ورودی نمونه ۱

```
Create_a_table_for Sepehr with_deposit_of 4000
Create_a_table_for Sepehr with_deposit_of 5500
Create_a_table_for Sepehr with_deposit_of 6000
Invitation_request_from Sepehr for Amin with_deposit_of 2000
Number_of_levels
```

```
Friends_of Amin
Users_on_the_same_level_with Amin
Introducer_of Amin
Number_of_users_in_level 0
Number_of_users_in_level 1
Credit_of Sepehr
Credit_of Amin
How_much_have_we_made_yet
End
```

خروجی نمونه ۱

```
Money is not enough
You now own a table
We already have a founder
User added successfully in level 1
2
No_friend
He_is_all_by_himself
Sepehr
1
1
1800
400
5300
```

Blockchain-2 (امتیازی)

- محدودیت زمان: ۲ ثانیه
- محدودیت حافظه: ۵۰ مگابایت

مهدیز به تازگی خیلی در Blockchain و Smart Contract خفن شده و به استخدام فردی به نام ایمانز در همین زمینه درآمده و هفته‌ای دست کم دو پروژه بلاک‌چین را در برنامه هفتگی خود دارد.

ایمانز که فردی جاه‌طلب است، می‌خواهد یک miner بخرد و کد آن را دست‌کاری کند تا بتواند کمی هم از شبکه بلاک‌چین پول‌شویی کند. از آن‌جا که خودش عرضه‌ی این کارها را ندارد آن را به مهدیز می‌سپارد.

بیش از این حرف نزنیم و بگیم، تو باید چه چیز هایی بدونی، و چه کاری از تو انتظار می‌ره:

از بدو خلقت تا یگانه لحظه اکنون که این متنو می‌خونی، m بلاک در شبکه ساخته شده که به صورت زنجیروار به هم متصل شده‌اند (blockchain). پیاده‌سازی محتوای block به طوری که متناسب سوال باشه بر عهده شماست. درون یک بلاک، n تراکنش (transaction) وجود دارد (ممکن است تعداد تراکنش های یک بلاک با بلاکی دیگر فرق کند!) که هر تراکنش متناظر با یک رشته است. هر یک از بلاک‌ها نیز رشته‌ای معرفی‌کننده و بسیار مهم به نام hash دارد که این رشته، تابعی از تمام تراکنش‌های موجود در بلاک و همچنین hash بلاک قبلی است. (البته ما در این سوال فرضیاتی برای ساده سازی مسئله انجام دادیم که در واقعیت به این شکل نیست. یعنی برای تولید hash بلاک iام، به تمام تراکنش‌های این بلاک و همچنین hash بلاک $i-1$ام نیاز داریم. (قطعا باید تا الان نگران تولید hash مربوط به بلاک شماره 1 خلقت شده باشی، نگران نباش اون hash رو ما به شما می‌دیم.)

اما چه ترکیب خاصی از تراکنش ها و hash قبلی، hash جدید را می‌سازد؟

ابتدا تابع `value(String s, int P)` را به صورت زیر تعریف می‌کنیم:

$$value(s, P) = ((\sum_{k=0}^{len(s)-1} s[k].P^k) \ mod \ 94) + 33$$

ورودی های تابع value:

- اول s که یک رشته است و $s[k]$ که کد ASCII نظیر به کاراکتر kام از آن رشته است که در محاسبات استفاده می‌شود.
- دوم P یک عدد اول منسوب به بلاک است؛ به این معنا که برای هر بلاک می‌تواند با بلاک دیگر فرق کند.

خروجی تابع value:

- خروجی عملیات ریاضی فوق منطقا یک عدد است که تابع value آن را به کارکتر نظیر به کد اسکی‌ای برابر همان عدد تبدیل می‌کند و return می‌کند. تضمین می‌شود این عدد قطعا کارکتر منطقی‌ای می‌سازد و در بازه‌ی معقولی از کد های ASCII قرار می‌گیرد. (به این فکر کن که چجوری تضمین می‌شه!)

تابع value دو جا استفاده می‌شود:

اول: برای بدست آوردن Primary String مربوط به بلاک. اینکه Primary String کجا استفاده می‌شه رو یکم دیگه بهت می‌گیم ولی اینکه چجوری این رشته رو برای یه block بسازی:

```
block.primaryString[i] = value(block.transactions[i], block.P)
```

همانطور که معلومه تابع value، کاراکتر nم از رشته Primary String یک بلاک را با عمل کردن روی تراکنش nم همان بلاک و با استفاده از عدد اول (P) همان بلاک را می‌سازد.

دوم: حال رسیدیم به **اصل کار** (ویلسون)، بدست آوردن hash بلاک. hash همون Primary String مربوط به بلاک است. فقط یه کاراکترش رو باید عوض کنی (جایی که hash بلاک قبلی استفاده می‌شه). سوالایی که باید واسه‌ت ایجاد شده باشه اینه: کدوم کاراکتر رو عوض کنم؟ جاش چی بذارم؟

ابتدا دومی رو جواب می‌دیم. جاش alter رو بذار:

```
alter = value(previousHash, block.P)
```

که مسلما previousHash همون هش بلاک قبلی است.

حالا سوال اولت، کارکتر alter رو بذار جای اندیس زیر:

$$alter\ \ mod\ \ block.n$$

که block.n تعداد تراکنش های همین بلاکیه ک داری تلاش می‌کنی hash اون رو حساب کنی.

ورودی

ما در ابتدا در یک خط به شما m را می‌دهیم که تعداد بلاک‌هایی‌ست که مونده رو دست‌مون. تضمین می‌کنیم که m عددی مثبت است.

خط بعد رو یک خط کامل بدون space بهت hash بلاک اول این دسته بلاک جامونده رو می‌دیم. پس از دادن hash بلاک اول $m-1$ بار اطلاعات بلاک های 2 تا m را به شما می‌دهیم. اطلاعات هر بلاک به صورت زیر است:

در یک خط به ترتیب ابتدا n و سپس P مربوط به بلاک داده می‌شود. یه تضمین دیگه هم می‌کنیم که P عدد اول است.

در نهایت در n خط بعدی، n رشته بدون space به عنوان رشته‌های تراکنش‌های 1 تا n مربوط به این بلاک به شما داده می‌شود.

خروجی

این‌جا hash مربوط به بلاک آخر (mام) رو تو یه خط چاپ کن.

مثال

ورودی نمونه ۱

```
2
abc
2 7
defghijklm
nopqrstuv
```

(از خط اول می‌فهمیم) ۲ بلاک داریم که (از خط دومش می‌فهمیم) hash بلاک اول که قول دادیم بهتون

بدیم abc است. مشخصات بلاک دوم به شکلی‌ست که اومده. ۲ تا تراکنش داره و عدد اولش هم $P = 7$ هستش. تو دو خط بعد اون هم رشته مربوط به تراکنش‌های اول و دومِ این بلاک اومده.

خروجی نمونه ۱

```
ny
```

بلاک آخر، بلاک دومه که اگر Primary Stringش رو بر اساس عدد اولش و تراکنش‌هاش حساب کنی، رشته بدست اومده، nw خواهد بود. نتیجه‌ی حاصل از پاس دادن hash قبلی (abc) به تابع value هم حرف y با کد اسکی 121 خواهد بود. $121\%2 = 1$ پس، از رشته پرایمری بدست آمده برای بلاک آخر، اندیس 1 را با y جای‌گذاری می‌کنیم که می‌شه همین hashای که بالا نوشته شده.

Machine Learning

شما اخیرا در شرکت گوگل استخدام شده‌اید. به تازگی، رسانه‌های حوزه‌ی تکنولوژی در سراسر دنیا، انتقادات متعددی به کیفیت اپلیکیشن‌های موجود در فروشگاه نرم‌افزاری گوگل $(GooglePlay)$ وارد کرده‌اند. به باور آن‌ها، گوگل به $feedback$ های کاربران گوش نمی‌دهد و بررسی کافی‌ای را روی محتوای اپلیکیشن‌ها پس از انتشار شدن اولیه‌ی آن‌ها نمی‌کند. به همین دلیل، توسعه‌دهندگان برخی از اپلیکیشن‌ها با سوء استفاده از این روند کاری گوگل، محتوای اپلیکیشن‌ها را پس از تایید شدن آن‌ها تغییر می‌دهند که مطلوب کاربران نیست و سبب اعتراض آن‌ها به عدم تطابق تعهدهای اپلیکیشن و کارکردهای آن و یا حتی در مواردی، کلاه‌برداری و سرقت اطلاعات توسط آن‌ها، خواهد شد.

مدیران گوگل به این نتیجه رسیده‌اند که کاربران $GooglePlay$ در صورت نارضایتی از اپلیکیشن‌ها، در اکثر مواقع اعتراض خود را به عنوان $review$ در صفحه‌ی اپلیکیشن ثبت می‌کنند. از این رو، آن‌ها پیشنهاد داده‌اند که تیمی وظیفه‌ی بررسی تک‌تک $review$ ها را به عهده گیرد. به سرعت، مدیران به این نتیجه رسیدند که به دلیل تعداد بسیار بالای این $review$ ها، عملا امکان بررسی دستی همه‌ی آن‌ها وجود ندارد. از این رو، تصمیم گرفتند نوشتن برنامه‌ای برای این منظور را به یک تیم فنی بسپرند.

تیمی که عضو آن شده‌اید، وظیفه‌ی این بررسی را به عهده‌ی شما گذاشته است. در این مرحله، لازم است برنامه‌ای بنویسید که یک $dataset$ شامل ۸۹۱ مورد $review$ نوشته شده - به همراه $polarity$ هر یک از آن‌ها در ستون دوم csv، که در صورت مثبت بودن $review$ برابر یک و در صورت منفی بودن آن، برابر صفر است، را دریافت کند. باید برنامه‌ی شما به کمک ۷۰۰ مورد اول، «یاد بگیرد» که چه $review$ هایی مثبت و چه $review$ هایی منفی هستند، و سپس این «مدل» یافته شده را روی باقی موارد (به جز ۷۰۰ مورد اول)، «تست» کند و نتیجه را در فایلی ذخیره کند. سپس، با مقایسه‌ی نتایج با فایل اصلی، اعلام کند که چند درصد (مثلا ۷۰ یا ۸۰ یا...) از $review$ ها، به درستی تخمین زده شده‌اند. توجه کنید که در فرآیند یادگیری نباید از موردی به غیر از ۷۰۰ مورد اول، استفاده شود.

برای دریافت دیتاست بر روی این لینک کلیک کنید.

- لازم به ذکر است که استفاده از کتاب‌خانه‌های آماده غیر مجاز است و سوال باید به زبان جاوا حل شود (در عمل مسائل این‌چنینی با پایتون حل می‌شوند، اما به دلیل سادگی الگوریتم مورد استفاده، پیاده‌سازی آن در جاوا نیز ممکن است). این سوال داوری خودکار ندارد و معیار مقایسه، درست تشخیص دادن $polarity$ درصد بالاتری از داده‌های $test$ خواهد بود. الگوریتم مورد

استفاده در سوال حتما باید از نوع $NaiveBayes$ باشد؛ توضیح این الگوریتم و آموزش آن را می‌توانید در اینترنت مشاهده کنید. در چند روز آینده، مستندی برای شرح این الگوریتم عرضه خواهد شد.

- می‌توانید برای بهبود عملکرد الگوریتم خود، از تکنیک‌های مختلف مانند حذف حروف پرتکرار که لزوما معنایی در تعیین $polarity$ ندارند (مانند the و نظایر آن)، حذف علائم نگارشی و کاراکترهای غیر مجاز و... استفاده کنید.

▼ **توجه**

دیتاست استفاده شده از این لینک بوده و مسابقه LoC و برگزارکنندگان آن، **هیچ‌گونه مسئولیتی در خصوص محتوای موجود در آن نمی‌پذیرند**.