زبان سی یا بهمن هاشمی؟

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

همه‌ی ما می‌دانیم که بهمن هاشمی همیشه دشمن قسم خورده‌ی برنامه‌نویس‌ها بوده و هست. او همیشه دوست دارد به هر نحوی که شده، آن‌ها را جلوی بقیه‌ی افراد جامعه ضایع کند. به همین دلیل در یک برنامه‌ی تلویزیونی، برای بی‌اهمیت نشان دادن کامپیوتر و برنامه‌نویسی، با حرکات جانانه‌ی دستان خود محاسبات مربوط به اعداد مختلط را حتی از برخی کامپیوترها هم سریع‌تر انجام می‌دهد! حال شما به عنوان دانشجوی درس مبانی برنامه‌سازی باید با حل کردن سوال زیر ثابت کنید توانایی زبان سی و کامپیوتر در محاسبه، بیشتر از بهمن هاشمی است و او را به سزای اعمالش برسانید!

یک حافظه با n خانه داریم که هر خانه‌ی آن قابلیت قرارگیری یک عدد مختلط را دارد و در هر لحظه در هر خانه‌ی دل‌خواه از حافظه ممکن است یک عدد مختلط وجود داشته باشد یا آن خانه خالی باشد. در ابتدا همه‌ی خانه‌های حافظه خالی هستند. سپس t خانه‌ی متمایز آن مقداردهی اولیه می‌شوند و پس از آن هم

q دستور داده می‌شود که دستورات به ترتیب داده شده باید روی حافظه اجرا شده و خروجی مناسب هر کدام داده شود. دقت کنید به ازای هر دستور اگر حداقل یکی از خطاهای مربوط به آن دستور رخ دهد دستور به طور کامل انجام نمی‌شود و هیچ تاثیری روی اطلاعات نمی‌گذارد.

تا انتهای سوال، به ازای هر عدد مختلط دل‌خواه به شکل $a + bi$، منظور از چاپ آن *به فرم مختلط*، به شکل جفت مرتب $(a,\ b)$ است.

تضمین می‌شود انتهای دستورات هر اندیس که از خانه‌های حافظه داده می‌شود قطعا در محدوده‌ی $[1, n]$ خواهد بود.

همچنین غیر از خطاهایی که در سوال پوشش داده شده‌اند، دستورات به فرم دیگری نخواهند بود و خطاهای دیگری نخواهند داشت.

در نهایت تضمین می‌شود به ازای هر عدد مختلط دل‌خواه مانند $a + bi$ که در دستوری طی اجرای برنامه داده می‌شود یا لحظه‌ای طبق عملیات موجود در دستورات تولید می‌شود، شروط زیر برقرار است:

$$-10^9 \leq a, b \leq 10^9$$

$$a, b \in \mathbb{Z}$$

دستورات به صورت زیر هستند:

حذف کردن عدد:

```
delete <i>
```

عدد مختلط موجود در خانه‌ی iم حافظه را در صورت وجود به فرم مختلط چاپ کرده و سپس آن خانه از حافظه را خالی می‌کند.

خطاها:

- در صورتی که خانه‌ی iم حافظه خالی بود پیغام زیر چاپ شود:

```
no number in <i>!
```

در صورت موفقیت‌آمیز بودن دستور، عددی که از حافظه پاک شده را به فرم مختلط چاپ کنید.

اضافه کردن عدد:

```
add <index> <a> <b>
```

در این دستور باید در صورت خالی بودن خانه‌ی $index$ام حافظه، عدد $a + bi$ در آن نوشته شود.

خطاها:

- اگر خانه‌ی $index$م حافظه خالی نبود عبارت زیر چاپ شود:

```
number currently exists in <index>!
```

در صورت موفقیت‌آمیز بودن دستور، پیغام زیر چاپ شود:

```
added successfully!
```

چاپ تعداد اعداد حافظه:

```
number count
```

در این دستور باید تعداد خانه‌های حافظه که خالی نیستند را چاپ کنید.

انجام عملیات:

```
operation <i> <op> <j> -> <k>
```

در این دستور باید عملیات مشخص شده در op روی اعداد موجود در خانه‌های i و j حافظه اعمال شود و حاصل در صورت امکان در خانه‌ی kم قرار بگیرد و در نهایت به فرم مختلط چاپ شود. در این‌جا op می‌تواند یکی از $+$، $-$ و * باشد.

خطاها:

- اگر حداقل یکی از خانه‌های iم و jم خالی بود پیغام زیر چاپ شود:

```
not enough operands!
```

- اگر خانه‌ی kم حاوی عدد بود و خالی نبود پیغام زیر چاپ شود:

```
destination currently full!
```

در غیر این صورت حاصل در خانه‌ی kم قرار می‌گیرد و همچنین به فرم مختلط چاپ می‌شود.

چاپ تعداد اعداد ناحیه‌ی مختصاتی:

```
count quadrant <i>
```

در این دستور باید تعداد اعداد در حافظه که در صفحه‌ی مختصات در ناحیه‌ی iم مختصاتی قرار می‌گیرند را چاپ کنید. دقت کنید اعدادی که روی یک محور یا مرکز مختصات قرار می‌گیرند جزء هیچ ناحیه‌ای به حساب نمی‌آیند. در این دستور تضمین می‌شود i یک عدد طبیعی و با شرط $1 \leq i \leq 4$ باشد.

چاپ ناحیه‌ی مختصاتی عدد:

```
quadrant point <i>
```

در این دستور باید بگویید عدد موجود در خانه‌ی iم حافظه در کدام ناحیه‌ی مختصاتی قرار دارد.

خطاها:

- اگر خانه‌ی iم حافظه خالی بود عبارت زیر چاپ شود:

```
no number in <i>!
```

- اگر عدد موجود در خانه‌ی iم روی یکی از محورها یا مرکز مختصات قرار داشت عبارت زیر چاپ شود:

```
in no quadrant!
```

در غیر این صورت شماره‌ی ناحیه‌ی مختصاتی که این عدد در آن قرار دارد چاپ شود.

تبدیل به مزدوج:

```
conjugate <i>
```

این دستور عدد موجود در خانه‌ی iم حافظه را در صورت وجود به مزدوج آن تبدیل می‌کند و مزدوجش را در قالب مختلط چاپ می‌کند.

مزدوج هر عدد مختلط دل‌خواه مانند $a+bi$، برابر با $a-bi$ است.

خطاها:

- اگر خانه‌ی iم حافظه خالی بود، پیغام زیر چاپ شود:

```
no number in <i>!
```

در غیر این صورت مزدوج این عدد چاپ شود و نیز در خانه‌ی iم حافظه قرار می‌گیرد.

دوران عدد مختلط در صفحه‌ی مختصات:

```
rotate <i> (clockwise)|(counterclockwise)
```

در این دستور عدد موجود در خانه‌ی iم حافظه در صورت وجود به اندازه‌ی 90 درجه در جهت ساعتگرد یا پادساعتگرد دوران یابد، به فرم مختلط چاپ شود و در نهایت در خانه‌ی iم حافظه قرار بگیرد.

خطاها:

- اگر عددی در خانه‌ی iم حافظه وجود نداشت پیغام زیر چاپ شود:

```
no number in <i>!
```

در غیر این صورت دوران روی عدد خانه‌ی iم انجام می‌شود و حاصل پس از چاپ، در خانه‌ی iم حافظه قرار می‌گیرد.

ورودی

در خط اول به ترتیب n (تعداد خانه‌های حافظه)، t (تعداد خانه‌های با مقدار اولیه) و q (تعداد دستورات) داده می‌شود.

$$1 \leq n, q \leq 10^3$$

$$1 \leq t \leq n$$

در t خط بعدی در هر خط یک خانه از حافظه به صورت زیر با مقدار اولیه‌ی آن داده می‌شود (بقیه‌ی خانه‌ها در ابتدا خالی هستند و تضمین می‌شود t خانه‌ی داده شده متمایز هستند):

$$index\ a\ b$$

که به این معنی است که در خانه‌ی $index$م حافظه عدد $a + bi$ قرار دارد. سپس در q خط بعدی در هر خط یک دستور به یکی از حالت‌های توضیح داده شده داده می‌شود.

خروجی

خروجی هر دستور را در قالب گفته شده چاپ کنید.

مثال

ورودی نمونه

```
10 4 15
2 1 3
3 4 -6
6 -11 0
9 -7 8
delete 2
add 3 -1 4
number count
operation 2 + 3 -> 5
operation 9 * 3 -> 2
quadrant point 2
number count
quadrant point 10
operation 3 - 9 -> 10
count quadrant 4
conjugate 1
quadrant point 9
```

```
conjugate 9
quadrant point 9
rotate 9 clockwise
```

خروجی نمونه

```
(1, 3)
number currently exists in 3!
3
not enough operands!
(20, 74)
1
4
no number in 10!
(11, -14)
2
no number in 1!
2
(-7, -8)
3
(-8, 7)
```

عروسک روسی

- محدودیت زمان: ۲ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

در دنیای مارموز ماتریکس، بهمن هاشمی از مجموعه‌ای از عروسک‌های روسی استفاده می‌کند تا مردم کره‌ی زمین را تحت سلطه‌ی خود نگه دارد. این عروسک‌ها نماد انسان‌هایی هستند که در ظاهر آزادند اما در واقع، هر کدام توسط دست‌های نامرئی بهمن هدایت می‌شوند. او با دقت آن‌ها را در آرایه‌ای پیچیده سازماندهی می‌کند، محتوای عروسک‌ها را تغییر می‌دهد و جابه‌جا می‌کند تا ذهن و رفتار مردم را کنترل کند. حالا شما باید وارد این بازی پیچیده شوید، با دستورات دقیق نظم این آرایه را حفظ کنید، و مراقب باشید. هر اشتباه ممکن است باعث فروپاشی کنترل او بر جهان شود!

در ابتدا یک آرایه از n عروسک روسی که **به ترتیب آیدی** پشت هم قرار گرفته‌اند داریم. همه‌ی عروسک‌ها **در ابتدا خالی هستند** اما در ادامه با استفاده از دستورات داده شده ممکن است عروسک‌ها درون یکدیگر قرار

بگیرند. وظیفه‌ی شما این است که عملیات مربوط به عروسک‌های روسی را دریافت و اجرا کنید.

دستورات

۱. قرار دادن چند عروسک در یک عروسک دیگر

بهمن که همیشه به دنبال کنترل پیچیده‌ترین سامانه‌ها و ذهن‌ها است، این دستور را به‌عنوان نمادی از ساختارهای درون‌درون در ماتریکس طراحی کرده است تا با قراردادن عروسک‌ها در یکدیگر، توانایی پنهان‌کردن و لایه‌بندی هر واقعیت را تا بی‌نهایت شبیه‌سازی کند. این روش به او کمک می‌کند تا همه‌چیز را زیر لایه‌های بی‌پایان از اطلاعات و توهمات بپوشاند؛ دقیقاً مثل نحوه‌ای که ذهن افراد در ماتریکس کنترل می‌شود!

با وارد شدن این دستور شما باید k عروسک داده شده را بردارید و در عروسک واقع در اندیس iم آرایه قرار دهید.

ورودی

```
put {k} {i}
{index of 1'st doll} {index of 2'st doll}...{index of k'st doll}
مثال:
put 2 12
8 4
```

دقت کنید تمام اندیس‌های ورودی، اندیس در آرایه هستند، نه id عروسک‌ها. تضمین می‌شود تمام k اندیس داده شده متمایز باشند و هیچ کدام برابر با i نباشند. تضمینی بر روی مرتب بودن این اندیس‌ها نیست و می‌توانند به هر ترتیبی باشند.

۲. خالی کردن عروسک

بهمن دستور خالی کردن عروسک‌ها را برای مواقعی طراحی کرده که می‌خواهد به سرعت تمام لایه‌های درونی سیستم را بررسی کند. او باور دارد که تنها با خارج کردن محتوای هر عروسک به ترتیب، می‌تواند به ذات اصلی و واقعیت پنهان هر سامانه یا انسان پی ببرد. بهمن از این دستور برای کشف حقیقت‌هایی که در پس چندین لایه از فریب و پیچیدگی مخفی شده‌اند، استفاده می‌کند؛ شبیه به افرادی که در ماتریکس، از دنیای توهم به دنیای واقعی می‌آیند.

با وارد شدن این دستور شما باید محتویات درون عروسک iم را از آن خارج کنید و به ترتیب id بعد از آن قرار دهید.

ورودی

```
empty {i}
:مثال
empty 12
```

دقت کنید تنها عروسک‌هایی که به طور مستقیم داخل عروسک iم قرار دارند بیرون می‌آیند. به طور مثال اگر دستور $empty\ 1$ داده شود و داخل عروسک اول آرایه، عروسک با آیدی x باشد که درون آن عروسک y قرار دارد، صرفاً عروسک x در اندیس دوم آرایه قرار می‌گیرد و y همچنان داخل x می‌ماند.

۳. نمایش تعداد عروسک‌های موجود در آرایه‌ی اصلی

بهمن، برای نظارت دقیق‌تر بر ماتریکس و اطمینان از کنترل کامل خود، این دستور را ایجاد کرده است تا تنها تعداد عروسک‌هایی که در آرایه‌ی اصلی باقی مانده‌اند را ببیند. این عروسک‌ها آن‌هایی هستند که به‌طور مستقل بیرون قرار دارند، نه آن‌هایی که درون یکدیگر قرار گرفته‌اند. با این روش، بهمن می‌تواند از وضعیت و تعداد دقیق "عروسک‌های خارجی" مطمئن باشد و بداند که چه تعدادی همچنان آزاد هستند.

با وارد شدن این دستور باید تعداد عروسک‌هایی که درون آرایه‌ی اصلی هستند را خروجی دهید. واضح است عروسک‌هایی که درون عروسک‌های دیگر قرار دارند در این شمارش لحاظ نمی‌شوند.

ورودی

```
main_dolls_count
```

خروجی

```
{number of dolls in main array}
```

۴. نمایش تعداد عروسک‌های موجود در یک عروسک

بهمن این دستور را طراحی کرده است تا بتواند به‌طور جامع و دقیق بر عروسک‌ها نظارت کند و کنترل خود را بر روی کل سامانه تقویت کند. با این دستور، او می‌تواند تعداد کل عروسک‌هایی که درون یک عروسک خاص قرار دارند را مشاهده کند. این اطلاعات به بهمن کمک می‌کند تا بفهمد هر عروسک چه تعداد از دیگر عروسک‌ها را درون خود نگه‌داری می‌کند و از این طریق عمق نفوذش را در ذهن‌ها و رفتارهای افراد در دنیای واقعی تحلیل کند. این نوع نظارت باعث می‌شود که او بتواند استراتژی‌های کنترلی خود را به‌طور مؤثرتری برنامه‌ریزی کند.

با وارد شدن این دستور باید تعداد تمامی عروسک‌هایی که درون عروسک iم هستند را خروجی دهید. در واقع این عمل باید به صورت بازگشتی انجام شود تا اگر عروسکی به صورت غیر مستقیم هم درون عروسک i م باشد، شمرده شود. خود عروسک iم را هم در این شمارش در نظر بگیرید.

ورودی

```
count {i}
:مثال
count 12
```

خروجی

```
{number of dolls in doll at index of i}
```

۵. نمایش آیدی یک عروسک

با وارد شدن این دستور شما باید id عروسک در جایگاه iم آرایه‌ی اصلی را چاپ کنید.

ورودی

```
get_id {i}
:مثال
get_id 4
```

خروجی

```
{the id of i_th doll in the main array}
```

ورودی

در خط اول n و q که به ترتیب تعداد عروسک‌ها و تعداد دستورات هستند وارد می‌شوند و در ادامه دستورات به ترتیب وارد می‌شوند.

$$1 \leq n \leq 1000$$

$$1 \leq q \leq 2000$$

مجموع k در تمام کوئری‌ها از ۱۰۰۰ بیشتر نمی‌شود. تضمین می‌شود در همه‌ی دستورها، تمام اندیس‌ها بین ۱ تا تعداد عروسک‌های موجود در آرایه در آن لحظه است و اندیس نامعتبری داده نخواهد شد.

خروجی

در خروجی به ازای هر دستور از نوع سوم، چهارم و یا پنجم جواب مناسب را در یک خط چاپ کنید.

مثال‌ها

ورودی نمونه ۱

```
5 6
put 2 2
1 4
get_id 1
count 1
main_dolls_count
empty 1
count 1
```

خروجی نمونه ۱

```
2
3
3
1
```

ورودی نمونه ۲

```
5 9
put 1 1
2
put 1 1
2
count 1
empty 1
count 1
count 2
get_id 1
get_id 2
get_id 3
```

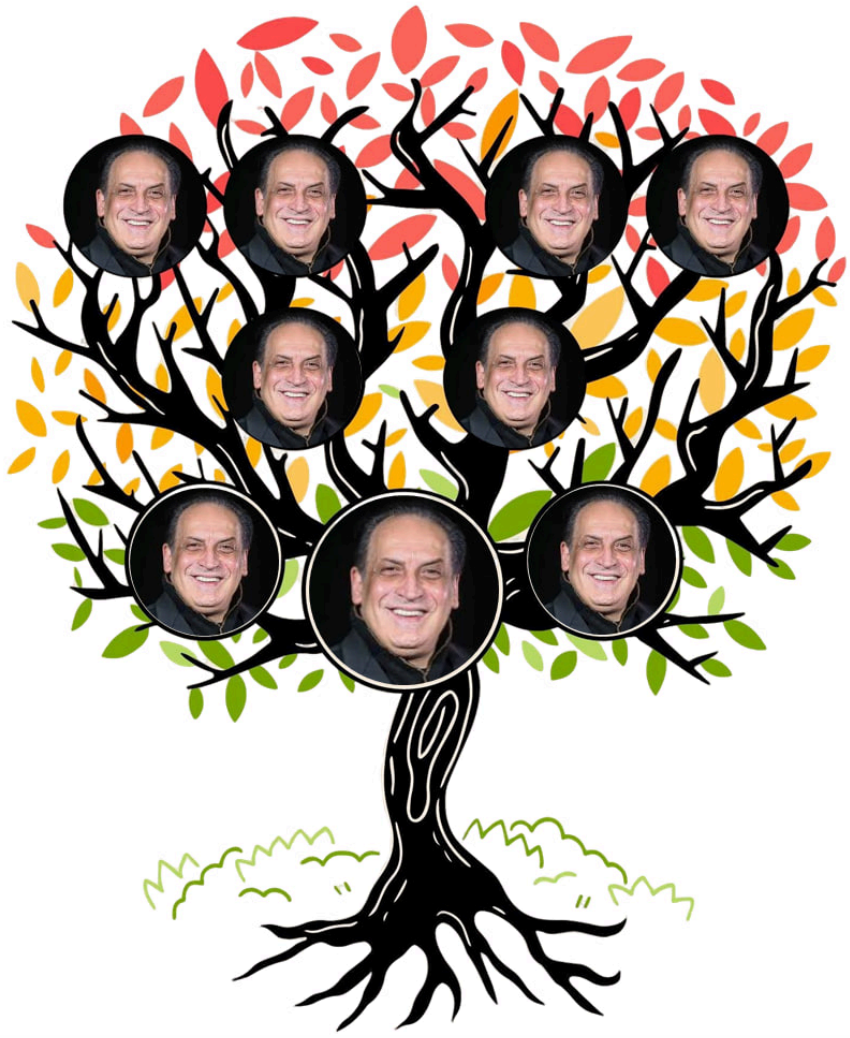
خروجی نمونه ۲

```
3
1
1
1
2
3
```

شجره‌نامه

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

بهمن هاشمی، مدیر مارموز ماتریکس، در تلاش است تا دنیای زیرین خود را با ایجاد یک شجره‌نامه‌ی پیچیده از انسان‌ها به تسلط خود درآورد. او با دقت در حال ثبت ویژگی‌های مختلف هر انسان از قد و رنگ چشم گرفته تا روابط خانوادگی است تا بتواند بر زندگی آن‌ها تأثیر بگذارد. این شجره‌نامه بهمن را قادر می‌کند تا نگاهی عمیق به ریشه‌های نسل‌ها داشته باشد و با جابه‌جایی‌ها و پیش‌بینی‌ها، به سلطه‌ی خود بر موجودات زنده ادامه دهد. اما او به کمک شما نیاز دارد تا این اطلاعات را به‌طور مؤثری مدیریت کند و به چالش‌های پیش‌رو پاسخ دهد.



در این سوال باید اطلاعات افراد موجود در شجره‌نامه‌ی بهمن را دریافت کنید و عملیات مدنظر او را انجام دهید. دقت کنید که برای ساده‌سازی تنها اطلاعات یکی از والدین هر فرد در شجره‌نامه وجود دارد (به عبارتی هر فرد در شجره‌نامه تنها فرزند یک نفر است).

دستورات

۱. اضافه کردن راس

هر فرد در شجره‌نامه با پنج ویژگی شناخته می‌شود: آیدی، اسم، آیدی پدر، قد، و رنگ چشم. تضمین می‌شود که هر آیدی منحصر به فرد است. چندین فرد والد ممکن است پدر نداشته باشند که در این صورت، آیدی پدر با ۱- مشخص می‌شود. آیدی اولین فرد هر عددی می‌تواند باشد اما تضمین می‌شود که آیدی افراد اعداد متوالی باشند.

$$0 \leq id \leq 2^{32}$$

$$0.00 < height < 3.00$$

$$TotalNumberOfPeople \leq 1000$$

برای افزودن افراد جدید، از دستور در قالب زیر استفاده می‌شود:

ورودی

```
add <id> <name> <parent-id> <height> <eye-color>
```

خطاها

اگر آیدی پدر وجود نداشته باشد و فرد از نخستین ساکنین نباشد، باید پیام زیر چاپ شود:

```
Parent doesn't exist!
```

۲. پیدا کردن نزدیک ترین جد مشترک

این دستور، نزدیک‌ترین جد مشترک دو فرد را پیدا می‌کند. اگر یکی از افراد جد دیگری باشد، آن فرد به عنوان جد انتخاب می‌شود.

ورودی

```
lca <first-node-id> <second-node-id>
```

خروجی

```
ID: <id> NAME: <name>
```

خطاها

اگر جد مشترکی وجود نداشته باشد:

```
No common ancestor exists!
```

۳. جابه‌جا کردن ۲ راس

در این دستور، اگر دو فرد جد یکدیگر نباشند، پدر آن‌ها جابجا می‌شود. اگر یکی از راس‌ها جد دیگری باشد یا هر دو از نخستین ساکنین شهر باشند، جابه‌جایی غیرممکن است.

در صورتی که یکی از راس‌ها جد دیگری نباشد، پدر دو راس را جابه‌جا می‌کنیم. توجه کنید که هنگام تغییر پدر یک راس، همه‌ی فرزندان آن راس نیز با آن جابه‌جا می‌شوند.

تضمین می‌شود که آیدی‌های داده شده درست هستند.

ورودی

```
switch <first-node-id> <second-node-id>
```

خطاها

- اگر یکی از راس‌ها جد دیگری باشد:

```
You can't switch with an ancestor!
```

- اگر دو راس از یک پدر باشند:

```
The given IDs are already siblings!
```

- اگر هر دو راس از نخستین ساکنان باشند:

```
The given IDs are both roots!
```

*تکته: *دقت کنید که یکی از راس‌ها می‌تواند از نخستین ساکنان باشد و در صورت جابه‌جایی با راسی دیگر، پدر آن راس پدر جدید آن می‌شود و راس دیگری بدون پدر می‌شود. اما هر دو نمی‌توانند از نخستین ساکنان باشند.

۴. چاپ کردن

در این دستور، درخت مربوط به فرزندان و نوادگان فرد به صورت درختی چاپ می‌شود. ترتیب چاپ کردن راس‌های هم‌ارتفاع بر اساس ترتیب آیدی است.هر ارتفاع با یک `\t` از ارتفاع قبلی جدا می‌شود.

ورودی

```
print <starting-node-id>
```

خروجی

```
starting-node
    child1
        child1.1
        child1.2
            child1.2.1
    child2
        child2.1
```

۵. پیش‌بینی فرزند

در این دستور، ویژگی‌های فرزند یک فرد با محاسبات وزنی پیش‌بینی می‌شود. دقت کنید که هدف تنها پیش‌بینی است و به شجره‌نامه چیزی اضافه نمی‌شود. برای این منظور نیاز به محاسبه‌ی پر احتمال‌ترین رنگ چشم، درصد احتمال آن و قد احتمالی فرد داریم؛ به طوری که با دور شدن از فرد مدنظر، احتمال تأثیرگذاری افراد کمتر می‌شود.

وزن هر فرد برابر با فاصله‌ی او از ریشه در درخت شجره‌نامه به علاوه‌ی یک است.

احتمال هر رنگ چشم به صورت جمع وزن‌های افراد دارنده‌ی آن رنگ چشم به روی جمع تمامی وزن‌ها به‌دست می‌آید. برای محاسبه‌ی قد نیز مانند بالا عمل می‌کنیم، اما در وزن هر فرد، قد او را نیز ضرب می‌کنیم و در نهایت جواب‌ها را با دقت دو رقم اعشار قطع می‌کنیم و نشان می‌دهیم. برای مثال اگر جواب ۱.۵۷۸ بود، عدد ۱.۵۷ را چاپ می‌کنیم.

دقت کنید که اگر احتمال دو رنگ چشم با یکدیگر برابر و بیشتر از بقیه شد، باید رنگی را خروجی دهید که از نظر الفبایی پیش از دیگری قرار دارد.

▼ مثال برای وزن

```
    1
   / \
  2   3
 / \
4   5
```

در درخت بالا وزن رأس ۱ برابر با ۱ و وزن رأس ۴ برابر با ۳ است.

▼ راهنمایی برای قطع کردن

با استفاده از تابع `floor` در کتابخانه‌ی `math.h` عبارات زیر را با `2lf.%` چاپ می‌کنیم.

$$\frac{\text{floor}(\text{Final Height} \times 100)}{100}$$

$$\frac{\text{floor}(\text{Eye Color Probability} \times 100)}{100}$$

ورودی

```
predict <node-id>
```

خروجی

```
eye color: "the most probable color" "probabality"%
height: "weighted average height"
```

۶. پایان

در صورت وارد شدن عبارت `end` اجرای برنامه متوقف میشود.

ورودی

دستورات مربوطه خط به خط در ورودی وارد می‌شوند تا اینکه در نهایت با وارد شدن دستور `end` برنامه متوقف شود.

خروجی

به ازای هر دستور خروجی مناسب را در صورت وجود چاپ کنید.

مثال

ورودی نمونه

```
add 1 Alice -1 1.8 Blue
add 2 Bob 3 1.6 Green
add 2 Bob 1 2.3 Green
add 3 Carol 1 1.1 Blue
add 4 Dave 2 1.9 Brown
add 5 Eve 2 1.7 Green
add 6 Frank 3 1.9 Blue
add 7 Jayce -1 1.8 Brown
print 1
lca 4 5
lca 4 6
switch 2 3
```

```
switch 5 6
print 1
predict 6
end
```

خروجی نمونه

```
Parent doesn't exist!
Alice
    Bob
        Dave
        Eve
    Carol
        Frank
ID: 2 NAME: Bob
ID: 1 NAME: Alice
The given IDs are already siblings!
Alice
    Bob
        Dave
        Frank
    Carol
        Eve
eye color: Blue 66.66%
height: 2.01
```

خرید و فروش در ماتریکس

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

توجه: این سوال دارای ۰.۳ نمره امتیازی بر روی تمرین است و قانون ۴۰ درصد بر روی آن اعمال نمی‌شود.

در تاریکی دنیای ماتریکس، بهمن هاشمی، مدیر مارموز و باهوش این سامانه‌ی کنترل، در سایه‌ها نشسته است و به ترفندهایی برای تسلط کامل بر زندگی انسان‌ها می‌اندیشد. او می‌داند که بازار، جایی است که قدرت واقعی در آن نهفته است. پس با یک لبخند شیطنت‌آمیز، تصمیم می‌گیرد یک سامانه‌ی نوین برای تبادل کالا میان کاربران و فروشگاه‌ها ایجاد کند؛ سامانه‌ای که به او این امکان را می‌دهد تا هر معامله را زیر نظر داشته باشد و کاربران را وادار به پیروی از نقشه‌های خود کند. در این دنیای پر از فریب و بازی‌های پنهان، او اطلاعات ارزشمند را جمع‌آوری کرده و در تاریک‌ترین زوایای مغز ماتریکس ذخیره می‌کند. این بار، او می‌خواهد با به دام انداختن کاربران در یک شبکه‌ی پیچیده از خرید و فروش، دنیای زیرزمینی ماتریکس را به تصرف خود درآورد. آیا کسی از خواب غفلت بیدار خواهد شد یا همه در دنیای فریب بهمن غرق خواهند شد؟

سامانه‌ی خرید و فروش دنیای ماتریکس به گونه‌ای است که کالاهای قابل معامله در آن با دستور مربوطه معرفی می‌شوند و کاربران و فروشگاه‌ها می‌توانند در آن حساب‌های مخصوص خود را بسازند و به معامله‌ی آن کالاها بپردازند. در ادامه، دستورات این سامانه در بخش‌های مختلف توضیح داده شده است.

توجه: در هرکدام از دستورات گفته شده اگر خطای خاصی از آن ذکر نشد تضمین می‌شود که در ورودی‌های آن دستور، آن خطای خاص وجود ندارد.

دستورات

اضافه کردن کالا به سامانه

```
add-good <good-name> <producer-price> <consumer-price> <tax-tag>
```

با دستور بالا، کالاهایی که می‌توانند میان فروشگاه‌ها و کاربرها رد و بدل شوند تعریف می‌شوند که در آن نام کالا، رشته‌ای متشکل از حروف کوچک انگلیسی است. قیمت تولید کننده و مصرف کننده هر دو اعداد صحیح و مثبت، و tax-tag برچسب مالیاتی هر محصول است که بر اساس آن هنگام خرید کالا از فروشگاه مقدار بیش‌تری هزینه از خریدار گرفته شده و مستقیما به عنوان مالیات توسط فروشگاه به سامانه‌ی مالیاتی پرداخت می‌شود. tax-tag مطابق جدول زیر به قیمت هر کالا در هنگام خرید می‌افزاید.

tag tax	percentage
low-priced	5
medium-priced	10
luxury	20

نکته: مالیات وارد بر کالا عددی صحیح است و در صورتی که در محاسبات عددی اعشاری به دست آمد با بزرگترین عدد صحیح کوچکتر از خود جایگزین می‌شود.

تضمین می‌شود در هیچ دستوری کالایی خارج از کالاهای تعریف شده با این دستور **نخواهد آمد.**

اضافه شدن فروشگاه به سامانه

```
register-store <id> <name> <password> <initial-credit> <k>
```

در این دستور پس از register-store هر فروشگاه، id، نام، کلمه‌ی عبور و موجودی اولیه‌ی حسابش و همچنین عدد صحیح و مثبت k وارد می‌شود و به سامانه به عنوان یک فروشگاه اضافه می‌شود. دقت کنید که نام و کلمه‌ی عبور فروشگاه هرکدام رشته‌ای متشکل از حروف کوچک و بزرگ انگلیسی و ارقام ۰ تا ۹ به طول حداکثر ۶۴ است. id و موجودی اولیه‌ی حساب، هر دو اعداد صحیح و مثبت شامل حداکثر ۹ رقم هستند.

- اگر فروشگاهی با id داده شده وجود داشت، خطای زیر نمایش داده می‌شود و به منوی اصلی برمی‌گردیم.

```
There exists a store with such id!
```

اگر خطای بالا رخ نداده بود، فروشگاه در ادامه در k خط متوالی لیست کالاهایی که در ابتدا موجود دارد را به همراه تعدادشان به صورت زیر وارد می‌کند.

```
<good-name> <amount>
```

پس از آن پیام زیر نمایش داده شده و فروشگاه با موفقیت به سامانه اضافه می‌شود. سپس به منوی اصلی باز می‌گردیم.

```
Store registered successfully!
```

اضافه شدن کاربر به سامانه

```
register-user <username> <password> <initial-credit>
```

در این دستور پس از register-user، کاربر، نام کاربری، کلمه‌ی عبور و موجودی اولیه‌ی حسابش را وارد می‌کند و به سامانه به عنوان یک کاربر اضافه می‌شود. دقت کنید که نام کاربری و کلمه‌ی عبور کاربر، هرکدام رشته‌ای متشکل از حروف کوچک و بزرگ انگلیسی و ارقام ۰ تا ۹ به طول حداکثر ۶۴ بوده و موجودی اولیه‌ی حساب، عددی صحیح و مثبت شامل حداکثر ۹ رقم است.

- اگر کاربری با username داده شده وجود داشت، خطای زیر نمایش داده می‌شود و به منوی اصلی برمی‌گردیم.

```
Username already exists!
```

در غیر این صورت کاربر به‌صورت موفقیت‌آمیز به سامانه اضافه می‌شود و پیام زیر نمایش داده می‌شود و به منوی اصلی باز می‌گردیم.

```
User registered successfully!
```

در ادامه امکانات هر یک از منوهای فروشگاه، کاربر و قابلیت‌های هرکدام از آن‌ها توضیح داده می‌شود.

منوی فروشگاه

برای وارد شدن به منوی فروشگاه در ابتدا باید از طریق منوی اصلی login کنیم. که به شرح زیر است.

لاگین کردن فروشگاه

```
login-store <id> <password>
```

خطاها: توجه کنید خطاها در صورت وجود به ترتیب گفته شده چاپ می‌شوند.

- اگر از نقش قبلی خارج نشده باشیم، باید خطای زیر نمایش داده‌شود و به منوی اصلی بازگردیم.

```
You should logout from current account to login to a new one!
```

- اگر فروشگاهی با id داده شده وجود نداشت پیغام زیر نمایش داده می‌شود و به منوی اصلی باز می‌گردیم.

```
No such store exists!
```

- اگر password وارد شده با password فروشگاه مطابقت نداشت پیغام زیر چاپ می‌شود و به منوی اصلی باز می‌گردیم.

```
Password doesn't match!
```

در صورتی که خطایی رخ نداد، فروشگاه به صورت موفقیت‌آمیز وارد حساب خود شده، پیام زیر نمایش داده می‌شود و به منوی فروشگاه وارد می‌شویم.

```
You're logged in as a store!
```

امکانات و قابلیت های منوی فروشگاه

نمایش موجودی

```
show-credit
```

با دستور بالا موجودی حساب فروشگاه را به صورت زیر نمایش می‌دهیم.

```
Credit: <store's credit>
```

خرید کالا

```
buy-goods <good-name> <amount>
```

با دستور بالا به تعداد مشخص شده کالا برای فروشگاه خریداری می‌شود. هزینه‌ی خرید کالا برای فروشگاه از producer-price محاسبه شده و شامل مالیات نمی‌گردد.

خطاها:

- اگر فروشگاه به میزان کافی موجودی نداشته باشد خطای زیر نمایش داده شود.

```
You don't have enough credit!
```

در غیر این صورت کالاها با موفقیت برای فروشگاه خریداری شده و پیغام زیر نمایش داده می‌شود.

```
Goods bought successfully!
```

نمایش سود

```
show-profit
```

با دستور بالا سود حاصل از فروش محصولات برای فروشگاه به صورت زیر نمایش داده می‌شود. فروشگاه به ازای فروش هر محصول به اندازه‌ی اختلاف consumer-price و producer-price سود می‌کند.

```
Profit: <store's profit>
```

منوی کاربر

برای وارد شدن به منوی کاربر در ابتدا باید از طریق منوی اصلی login کنیم که به شرح زیر است.

لاگین کردن کاربر

```
login-user <username> <password>
```

- اگر از نقش قبلی لاگ‌اوت نکرده باشیم، باید خطای زیر نمایش داده شود و به منوی اصلی باز می‌گردیم.

```
You should logout from current account to login to a new one!
```

- اگر کاربری با username داده شده وجود نداشت، پیغام زیر نمایش داده می‌شود و به منوی اصلی باز می‌گردیم.

```
No such username exists!
```

- اگر password وارد شده با password کاربر مطابقت نداشت پیغام زیر چاپ می‌شود و به منوی اصلی باز می‌گردیم.

```
Password doesn't match!
```

در غیر این صورت کاربر به صورت موفقیت‌آمیز وارد حساب خود شده، پیام زیر نمایش داده می‌شود و به منوی کاربر وارد می‌شویم.

```
You're logged in as a user!
```

امکانات و قابلیت‌های منوی کاربر

نمایش موجودی

```
show-credit
```

با دستور بالا موجودی حساب کاربر را به صورت زیر نمایش می‌دهیم.

```
Credit: <user's credit>
```

افزایش موجودی

```
charge <amount>
```

با دستور بالا مبلغ مشخص شده به موجودی حساب کاربر افزوده می‌شود و پیغام زیر نمایش داده می‌شود.

```
Your account successfully charged to <total-credit>!
```

نمایش کالاهای خریداری شده

```
show-stuff
```

با دستور بالا تمام کالاهای خریده شده توسط کاربر به همراه فروشگاهی که از آن خریداری شده به ترتیب زمان خرید نمایش داده می‌شوند. اگر کالاها مربوط به یک خرید بودند به ترتیب اضافه شدن به سبد خرید در

خطوط متوالی در قالب زیر نمایش داده می‌شوند.

```
<good-name> <store-name>
```

- اگر تا آن زمان کالایی توسط کاربر خریداری نشده بود، پیغام زیر نمایش داده شود.

```
You haven't done any transactions yet!
```

ایجاد سبد خرید جدید

```
create-shopping-cart <store-id>
```

با دستور بالا سبد خریدی خالی از محصولات فروشگاه مشخص شده ساخته می‌شود.

نکته: سبد خریدهای هر کاربر به صورت پشته ذخیره می‌شوند و کاربر برای دستورهای اضافه و حذف و خرید کالا تنها به جدیدترین سبد خریدی که ساخته دسترسی دارد و برای دسترسی به سبد خریدهای قبلی باید همه‌ی سبد خریدهای جدیدتر را یا حذف کرده و یا نهایی کند.

- اگر فروشگاه با id مشخص شده وجود نداشته باشد، خطای زیر نمایش داده شود.

```
There's no store with such id!
```

در غیر این صورت سبد خرید کالاها با موفقیت ایجاد شده و پیغام زیر نمایش داده می‌شود.

```
New shopping cart added successfully!
```

اضافه کردن کالا به سبد خرید

```
add-to-shopping-cart <good-name> <number>
```

با دستور بالا به تعداد مشخص شده از کالای مشخص شده به آخرین سبد خرید کاربر اضافه می‌شود.

- اگر کاربر تا آن زمان سبد خریدی نساخته باشد، باید خطای زیر نمایش داده شود.

```
No shopping cart created yet!
```

- اگر فروشگاه مشخص شده برای **آخرین سبد خرید** از کالای مشخص شده نداشت یا تعداد کالاهایش کم‌تر از مقدار خواسته شده بود، خطای زیر نمایش داده شود.

```
Not enough goods!
```

در غیر این صورت کالا به تعداد خواسته شده به آخرین سبد خرید کاربر اضافه شود و پیغام زیر نشان داده شود.

```
<number> of <good-name> added to your shopping cart successfully!
```

خالی کردن سبد خرید

```
clear-shopping-cart
```

دستور بالا تمام محتوای آخرین سبد خرید کاربر را پاک می‌کند.

- اگر کاربر تا آن زمان سبد خریدی نساخته باشد، خطای زیر نمایش داده شود.

```
No shopping cart created yet!
```

- اگر سبد خرید در حال حاضر خالی باشد خطای زیر نمایش داده شود.

```
Shopping cart is already empty!
```

در غیر این صورت محتوای آخرین سبد خرید پاک شده و پیغام زیر نمایش داده شود.

```
Shopping cart cleared successfully!
```

نمایش محتوای سبد خرید

```
show-shopping-cart
```

با دستور بالا محتوای آخرین سبد خرید کاربر در قالب زیر چاپ می‌شود. کالاها به ترتیب اضافه شدنشان در خطوط متوالی نشان داده شوند.

```
Shopping cart from store <store-id>:
    <good-name> <price>
```

قیمت مقابل هر کالا، consumer-price بدون درنظر گرفتن مالیات وارد بر آن است.

- اگر کاربر تا آن زمان سبد خریدی نساخته باشد، خطای زیر نمایش داده شود.

```
No shopping cart created yet!
```

- اگر آخرین سبد خرید خالی باشد، خطای زیر نمایش داده شود.

```
Shopping cart is empty!
```

حذف کردن سبد خرید

```
remove-shopping-cart
```

دستور بالا آخرین سبد خرید کاربر را از لیست سبد خریدها حذف می‌کند.

- اگر کاربر تا آن زمان سبد خریدی نساخته باشد، خطای زیر نمایش داده شود.

```
No shopping cart created yet!
```

در غیر این صورت آخرین سبد خرید با موفقیت حذف شده و پیغام زیر نمایش داده شود.

```
Shopping cart removed successfully!
```

نهایی کردن سبد خرید

```
buy-shopping-cart
```

با دستور بالا آخرین سبد خرید کاربر از فروشگاه مربوطه خریده می‌شود.

تذکر: مبلغ هزینه شده برای نهایی کردن سبد خرید از رابطه‌ی زیر محاسبه می‌شود:

$$\sum_{i=1}^{n} \lfloor \text{consumer-price}_i \times (1 + \text{tax}_i) \rfloor \times \text{amount}_i$$

- اگر کاربر تا آن زمان سبد خریدی نساخته باشد، خطای زیر نمایش داده شود.

```
No shopping cart created yet!
```

- اگر در سبد خرید کالایی موجود نباشد، خطای زیر نمایش داده شود.

```
Shopping cart is empty!
```

- اگر فروشگاه از کالایی به تعداد خواسته شده موجود نداشته باشد، خطای زیر نمایش داده شود.

```
There's not enough of chosen goods in store <store-id>!
```

- اگر با درنظر گرفتن مالیات، موجودی حساب کاربر کم‌تر از مبلغ سبد خرید باشد، خطای زیر نمایش داده شود.

```
You don't have enough credit!
```

در غیر این صورت کاربر با موفقیت کالاها را از فروشگاه می‌خرد و پیغام زیر نمایش داده می‌شود. سبد خرید پس از نهایی شدن خرید از لیست سبد خریدها حذف می‌شود.

```
You bought the shopping cart successfully!
```

نمایش آخرین تراکنش

```
show-last-transaction
```

با دستور بالا جزئیات آخرین خرید نهایی‌شده‌ی کاربر در قالب زیر نمایش داده می‌شود.

```
<number-of-goods> good(s) from store <store-name> were bought!
Total cost: <goods-total-cost>
Total tax: <total-tax>
```

- اگر تا آن زمان خریدی انجام نشده بود، خطای زیر نمایش داده شود.

```
You haven't done any transactions yet!
```

دستورات کلی

در اینجا دستوراتی را بیان می‌کنیم که در همه‌ی منوها وجود دارند.

نمایش کالاهای یک فروشگاه

```
show-store-stuff <store-id> price|amount|alphabetically
```

با دستور بالا تمام کالاهای فروشگاه مشخص شده به ترتیب ویژگی خواسته شده در خط‌های متوالی در قالب زیر نمایش داده می‌شوند.

تکته: برای ویژگی‌های amount و price به صورت نزولی مرتب شوند.

```
<good-name> <amount> <consumer-price> <tax-tag>
```

در صورت مرتب سازی برحسب قیمت (تعداد) اگر قیمت (تعداد) دو کالا برابر بود آن دو را به ترتیب حروف الفبا مرتب می‌کنیم.

- اگر فروشگاه با id مشخص شده وجود نداشت، خطای زیر نمایش داده می شود.

```
There's no store with such id!
```

- و اگر فروشگاه مشخص شده کالایی موجود نداشت، پیغام زیر نمایش داده می شود.

```
The store is empty!
```

خروج از حساب

```
logout
```

با زدن این دستور در هر نقشی که باشیم (فروشگاه یا کاربر) از آن خارج می‌شویم و به منوی اصلی باز می‌گردیم و پیام زیر نمایش داده می‌شود.

```
You're logged out!
```

- اگر در حساب کاربر یا فروشگاهی نباشیم، پیغام زیر نمایش داده می‌شود.

```
You're not logged in any account!
```

خارج شدن از برنامه

```
exit
```

با زدن این دستور در هر منو و یا نقشی که باشیم، برنامه به پایان می‌رسد.

توجه: در این سوال مجاز به استفاده از qsort نیستید.

ورودی

هرخط از ورودی شامل یک دستور است و تا زمانی که exit نیاید، ورودی ادامه پیدا می‌کند.

خروجی

در خروجی عبارات توضیح داده شده چاپ می‌شوند.

مثال‌ها

ورودی نمونه ۱

```
add-good pizza 8 10 medium-priced
register-store 1 restaurant passres 100 1
pizza 100
register-user bahman hAshEmi20 30
login-user bahman hAshEmi20
create-shopping-cart 1
buy-shopping-cart
add-to-shopping-cart pizza 5
charge 30
buy-shopping-cart
logout
exit
```

خروجی نمونه ۱

```
Store registered successfully!
User registered successfully!
You're logged in as a user!
New shopping cart added successfully!
Shopping cart is empty!
5 of pizza added to your shopping cart successfully!
Your account successfully charged to 60!
You bought the shopping cart successfully!
You're logged out!
```

ورودی نمونه ۲

```
add-good matrixCalculator 50 80 luxury
add-good russianDoll 10 15 low-priced
register-store 10 matrixShop goodPassword 500 1
matrixCalculator 30
register-user notBahman hAshEmI1234 200
login-store 10 goodPassword
show-profit
```

```
logout
add-good familyTree 30 50 medium-priced
login-store 10 goodPassword
buy-goods familyTree 15
logout
login-user notBahman hAshEmI1234
create-shopping-cart 10
add-to-shopping-cart familyTree 2
add-to-shopping-cart matrixCalculator 1
buy-shopping-cart
charge 150
buy-shopping-cart
show-stuff
logout
login-store 10 goodPassword
show-profit
logout
exit
```

خروجی نمونه ۲

```
Store registered successfully!
User registered successfully!
You're logged in as a store!
Profit: 0
You're logged out!
You're logged in as a store!
Goods bought successfully!
You're logged out!
You're logged in as a user!
New shopping cart added successfully!
2 of familyTree added to your shopping cart successfully!
1 of matrixCalculator added to your shopping cart successfully!
You don't have enough credit!
Your account successfully charged to 350!
You bought the shopping cart successfully!
familyTree matrixShop
familyTree matrixShop
matrixCalculator matrixShop
You're logged out!
You're logged in as a store!
```

```
Profit: 70
You're logged out!
```