ماشین‌حساب خرکی!

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

خره می‌خواهد نامه‌ای به اژدها بنویسد و او را به مرداب دعوت کند. پس از کلنجار رفتن بسیار با مغز نداشته‌اش، خره متنی نوشته است و با کلی ذوق و شوق آن را به شرک نشان می‌دهد. شرک پس از خواندن نامه، به خر بودن او پی می‌برد و تصمیم می‌گیرد تا به دوستش در نوشتن این نامه کمک کند.

شرک در ابتدا متوجه می‌شود که خره به دلیلی اعداد نامه را به زبان خرها که بسیار نزدیک به زبان رومی است، نوشته و در تمامی محاسبات نیز اشتباه کرده است. او که نگران می‌شود تا خره ساعت و تاریخ اشتباهی را به اژدها اعلام کند، دست به ساخت ماشین حسابی می‌زند که بتواند محاسبات را انجام دهد و همچنین اعداد را به اعداد عادی تبدیل کند.

در این سوال شما باید یک ماشین‌حساب رومی طراحی کنید که بتواند پنج عمل اصلی یعنی جمع، تفریق، ضرب، تقسیم و باقی‌مانده‌گیری را انجام دهد. تفاوت این ماشین‌حساب با ماشین‌حساب‌های معمولی این است که اعداد ورودی آن به صورت اعداد رومی داده می‌شوند. عملیات نیز طبق جدول زیر مشخص می‌شود:

عملیات مورد انتظار	ورودی داده‌شده
+	plus
-	minus
*	times
/	divided by
%	mod

دقت کنید در عملیات تقسیم، قسمت صحیح خارج‌قسمت را خروجی دهید.

برای آشنایی با قانون و نحوه‌ی ساخت اعداد رومی می‌توانید به این لینک مراجعه کنید.

ورودی

در خط اول ورودی عدد طبیعی n می‌آید که نشان‌دهنده‌ی تعداد عملیات است.

$$1 \leq n \leq 100000$$

سپس در هر یک از n خط بعدی، یک رشته در قالب زیر داده می‌شود:

```
roman_number1 operation roman_number2
```

که در آن `roman_number1` و `roman_number2` دو عدد رومی هستند. (تضمین می‌شود که اعداد رومی با کاراکترهای بزرگ داده شوند). همچنین `operation` یکی از رشته‌های جدول بالاست.

$$1 \leq roman\_number1 \leq 50$$

$$1 \leq roman\_number2 \leq 50$$

خروجی

در خروجی حاصل هر عمل را در یک خط چاپ کنید.

مثال‌ها

برای درک بهتر، به مثال‌های زیر توجه کنید.

ورودی نمونه ۱

```
3
II plus V
X times I
IV plus V
```

خروجی نمونه ۱

```
7
10
9
```

ورودی نمونه ۲

```
2
XX divided by III
XII mod V
```

خروجی نمونه ۲

```
6
2
```

کمی privacy لطفا

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

خره قبل از ارسال نامه برای اژدها، خبرهایی مبنی بر جاسوسی لرد فارکواد درباره‌ی تمامی نامه‌های ارسالی و دریافتی می‌شنود. لرد فارکواد خبیث که با شکنجه‌ی مرد زنجبیلی به اطلاعاتی که می‌خواسته دست نیافته است، متن تمامی نامه‌ها را مو به مو می‌خواند تا مطمئن شود نکته‌ای از نظرش مخفی نمی‌ماند.

خره که از خوانده شدن نامه‌ی خصوصی‌اش توسط لرد خبیث بدش می‌آید، پس از مشورتی که با شرک می‌کند متوجه می‌شود که باید روشی برای رمزنگاری نامه‌ی خود پیدا کند. علاوه بر این، باید مطمئن شود که لرد در نامه‌اش تغییری ایجاد نکرده باشد و محتوای نامه به درستی به اژدها منتقل شود.

شرک به خره می‌گوید که بهتر است از عملگر XOR برای $Scramble$ و $Discramble$ کردن متن‌ها استفاده کند و البته با توجه به این‌که ممکن است متن‌های $scramble$ شده با این روش، حاوی کاراکترهای غیرقابل نمایش روی نامه باشند، آن‌ها را به‌صورت $Hexadecimal$ نمایش می‌دهند.

همچنین، با توجه به این‌که ممکن است لرد فارکواد در انتقال پیام‌ها اخلالی ایجاد کند، یا کلیدها تغییر یابند، باید از این موضوع مطلع شوند و از $Checksum$ برای اطمینان از صحت فرآیند استفاده کند. یعنی بدین‌گونه که در انتهای پیامِ رمزنگاری شده، ۲ بایت $Checksum$ قرار می‌گیرد. نحوه‌ی محاسبه‌ی این $Checksum$ در ادامه توضیح داده می‌شود.

همان‌طور که انتظار می‌رود، خره به کل متوجه صحبت‌های شرک نمی‌شود. حال شرک باید برنامه‌ای بنویسد تا خره با داشتن کلید (key) بتواند نامه را رمزگذاری کرده و به اژدها ارسال کند.

نکته‌ها

روش Scramble کردن

فرض کنیم طول کلید k باشد و مقدار عددی کاراکتر iم آن را نیز a_i بنامیم؛ متن ($message$) را به واحدهای kتایی تقسیم می‌کنیم و هر کاراکتر را با کاراکتر نظیر کلید، XOR میکنیم. به زبان ریاضی اگر کاراکتر iم متن ورودی m_i و iمین عدد اسکرمبل شده را s_i بنامیم، داریم:

$$s_i = m_i \oplus a_{(i\%k)}$$

در صورت لزوم می‌توانید از طریق این صفحه خودتان این شیوه را آزمایش کنید.

نحوه‌ی محاسبه و قرارگیری Checksum

شما باید کد ASCII کل کاراکترهای پیام اصلی (**رمزنگاری نشده**) را با هم جمع کنید و در نهایت، کد هگزادسیمالِ ۲ بایتِ وزنِ پایین‌تر آن که ۴ کاراکتر می‌شود را به انتهای پیام اضافه کنید. توجه کنید که این دو بایت باید به‌صورت Little Endian بیایند؛ بعداً با مفهوم Endianness بیشتر آشنا خواهید شد؛ ولی منظور از آن در این سوال، یعنی بایتی که زودتر می‌آید، بایتی است که وزن کمتری دارد و سپس بایت با وزن بالاتر می‌آید. برای مثال، عدد `0x8C30` که یک عدد ۱۶ بیتی (۲ بایتی) است، به‌صورت Little Endian، به شکل `0x30 0x8C` می‌آید.

اگر متوجه نشدید، نگران نباشید! در مثال زیر به‌طور کامل متوجه خواهید شد.

بررسی یک مثال

فرض کنید کلید، رشته‌ی `Bitkey123` باشد و می‌خواهیم پیام زیر را رمزگذاری کنیم.

```
Hello World! How Are You?
```

با توجه به اینکه طول رشته‌ی کلید برابر با ۹ است، پیام را به بخش‌های با طول ۹ می‌شکنیم و هر بخش را کاراکتر به کاراکتر با کلید XOR می‌کنیم. همان‌طور که گفتیم، با توجه به وجود کاراکترهای غیرقابل نمایش، آن را به صورت Hexadecimal نمایش می‌دهیم.

```
"Hello Wor" XOR "Bitkey123" -> Hex: 0A 0C 18 07 0A 59 66 5D 41
"ld! How A" XOR "BitKey123" -> Hex: 2E 0D 55 4B 2D 16 46 12 72
"re You?"   XOR "Bitkey123" -> Hex: 30 0C 54 32 0A 0C 0E
```

محاسبه Checksum

```
(ASCII('H') + ASCII('e') + ... + ASCII('u') + ASCII('?')) & 0xFFFF
```

در این مثال، مجموع کد ASCII کاراکترهای سازندهٔ پیام، برابر عدد ۲۱۴۳ و معادل `0x085F` است؛ بنابراین طبق Little Endian باید انتهای خروجی، `5F08` اضافه کنیم. دقت کنید که اگر عدد حاصله، بیشتر از ۲ بایت (۱۶ بیت) بود، باید دو بایت وزنِ پایین‌تر آن در نظر گرفته شود.

ساخت خروجی:

هگزادسیمال‌های به‌دست آمده را کنار هم قرار می‌دهیم، و Checksum محاسبه شده را به‌طور صحیح، در انتها قرار می‌دهیم:

```
0a0c18070a59665d412e0d554b2d16461272300c54320a0c0e5f08
```

خب؛ نکته‌ای که در استفاده از Checksum وجود دارد چیست؟

در هنگام تبدیل رشته‌ی کدشده به متن اولیه، با توجه به خاصیت تقارنی XOR، کد هگزادسیمال را به رشته تبدیل می‌کنیم، سپس رشته‌ی حاصل (به‌جز دو بایت آخر آن که در واقع Checksum است) را مانند بالا با رشته‌ی کلید XOR می کنیم تا رشته‌ی اولیه به‌دست آید؛ **سپس بررسی می‌کنیم که آیا Checksum رشته‌ی اولیه‌ی به‌دست آمده، با Checksum ورودی یکسان است یا خیر. اگر یکسان نبود، پی می‌بریم که یا کلید اشتباه وارد شده یا اختلالی در تبادل داده صورت گرفته است.**

دستورات

در این برنامه شما باید بتوانید هر یک از ۳ دستور `<encrypt with <key>` ، `decrypt with <key` و `exit` را پردازش کنید. توضیحات آن‌ها در ادامه آمده‌اند:

- **دستور encrypt**

```
encrypt with <key_string>
<message>
```

  - با دریافت این دستور طبق فرمت بالا، در همان خط، کلید داده می‌شود؛ و در خط دوم نیز متن پیامی که باید رمزنگاری شود داده می‌شود. تضمین می‌شود message شامل `\n` نیست و طول آن حداکثر ۵۰۰ کاراکتر است.
  - در خروجی این دستور باید نمایش lowercase hexadecimal متنِ scramble شده را در یک خط چاپ کنید.
  - لازم به ذکر است که طول رشتهٔ `<key_string>` بیشتر از ۱۰۰ کاراکتر نیست.
- **دستور decrypt**

```
decrypt with <key_string>
<encrypted_message>
```

  - با دریافت این دستور طبق فرمت بالا، در همان خط، کلید داده می‌شود؛ و در خط بعدی، متن پیام رمزنگاری‌شده داده می‌شود. دقت کنید که کد Hex ورودی ممکن است شامل حروف بزرگ یا کوچک باشد. تضمین می‌شود که کد هگزادسیمال ورودی معتبر است. طول رشتهٔ ورودی بیشتر از ۱۰۰۰ کاراکتر نیست.
  - در خروجی باید متن discramble شده‌ی پیام با کلید را در یک خط چاپ کنید.
  - **در صورتی که پی بردید که کلید غلط است، باید در خروجی عبارت `Invalid key!` را چاپ کنید.**
  - لازم به ذکر است که طول رشته‌ی `<key_string>` بیشتر از ۱۰۰ کاراکتر نیست.
- **دستور exit**

```
exit
```

با دریافت این دستور از برنامه خارج می‌شویم.

نکته: ابتدا و انتهای دستورها ممکن است Whitespace اضافه بیاید.

مثال

ورودی نمونه

```
encrypt with DragonKey2024
Dear Dragon, The kingdom is in grave danger!
decrypt with Bitkey123
230c09052f0b7d56230c0a1f4c3a4b561d0a594c1b07470e3d0c594e18f7185f08
encrypt with Testkey
We need your help to defeat him before it's too late. Beware of his traps a
decrypt with Bitkey123
1108020e450d59571329001a0c01165c12553006194b11115412573008l3040b5e421244300
exit
```

خروجی نمونه

```
001700154f2a39041e5d5e1e14101a0447040725021d5d5d125d375208094f0939040f57105
Invalid key!
0300531a0e001d741c1c01194511310903541f0a59300015110a11593c0c1e5409001f3b171
Save the kingdom from the dragon's wrath!
```

شطرنج

- محدودیت زمان: ۲ ثانیه
- محدودیت حافظه: ۵۱۲ مگابایت

خره پس از هزاران بدبختی و هرطور که شده، نامه را ارسال می‌کند. چند روز بعد در جواب، نامه‌ای از طرف اژدها دریافت می‌کند که می‌گوید فردا به مرداب خواهد آمد. اینجاست که خره متوجه نکته مهمی می‌شود، او فکر نکرده است که اژدها را چگونه سرگرم کند!

پس از زیر و رو کردن خانه شرک، او به شطرنج جادویی‌ای برخورد می‌کند.

مهره‌های سفید در یک طرف بودند و مهره‌های سیاه در طرف دیگر. ولی این بار به جز شاه و وزیر و رخ، مهره‌های جدید و جادویی هم اضافه شده بودند. مثل اژدهاها، نینجاها، و حتی جادوگرها! هر مهره با یک حرف جادویی نمایش داده می‌شد و اگر سفید بود، با حرف کوچک و اگر سیاه بود، با حرف بزرگ ظاهر می‌شد.

خره متوجه می‌شود که در این شطرنج، باید با بیان کردن دستورات مهره‌ها را حرکت داد. او که باید در طول یک روز بازی را یاد بگیرد تا آبرویش جلوی اژدها نرود، به شرک التماس می‌کند که برنامه‌ای بنویسد تا حالات مختلف صفحه را پس از اعمال حرکات مختلف به او نشان دهد.

مهره‌های این شطرنج در جدول اسرارآمیز زیر آورده شده است:

نام مهره	کاراکتر
king	k
queen	q

نام مهره	کاراکتر
rook	r
bishop	b
knight	n
pawn	p
cannon	c
chariot	h
sorcerer	s
dragon	d
ninja	i
warrior	w
mage	m
golem	g

در طی سوال دو نوع دستور داریم:

- دستورات اولیه: به فرم `{color} {piece} is at column {x} row {y}` هستند و به این معنا که مهرهٔ `piece` و رنگ `color` (که یا `white` است یا `black`) باید در ردیف `x` و ستون `y` قرار گیرد.
- دستورات ثانویه: به فرم `{color} {piece} to column {x} row {y}` هستند و باید مهرهٔ موردنظر (در صورت وجود) جایگزین مهره‌ای شود که در خانهٔ ذکرشده قرار دارد.

ورودی

در خط اول دو عدد `m` و `n` با یک فاصله از هم داده می‌شود. در خط دوم عدد `i` داده می‌شود که نشان‌دهندهٔ تعداد دستورات نوع اول است. در `i` خط بعدی دستورات اولیه داده می‌شوند. سپس در خط

بعدی عدد `p` داده می‌شود که تعداد دستورات ثانویه است و در خطوط بعدی می‌آید.

$$1 \leq n, m \leq 50$$

$$1 \leq i \leq \min(28, n \times m)$$

$$0 \leq p \leq 100$$

خروجی

صفحهٔ شطرنج را در `m` ردیف و `n` ستون چاپ کنید. اگر در خانه‌ای مهرهٔ سفید قرار دارد کاراکتر مربوط به آن را به صورت کوچک و اگر مهرهٔ سیاه قرار داشت به صورت بزرگ چاپ کنید. در غیر این صورت `.` چاپ کنید.

مثال

ورودی نمونه ۱

```
5 34
3
black chariot is at column 13 row 2
white warrior is at column 1 row 5
black dragon is at column 10 row 3
0
```

خروجی نمونه ۱

```
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . H . . . . . . . . . . . . . . . . .
. . . . . . . . . D . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
w . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

ورودی نمونه ۲

```
6 6
5
black queen is at column 1 row 1
white queen is at column 1 row 2
black king is at column 3 row 3
white king is at column 4 row 4
white pawn is at column 5 row 5
5
black queen to column 1 row 2
white king to column 1 row 1
black sorcerer to column 1 row 1
white pawn to column 1 row 3
black king to column 3 row 3
```

خروجی نمونه ۲

```
k . . . . .
Q . . . . .
p . K . . .
. . . . . .
. . . . . .
. . . . . .
```

گفتی چاپ؟

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

پس از سر و کله زدن با شرک بر سر موضوعات مختلف نگارشی، خره با استفاده از برنامه‌ای که شرک نوشت، متن خود را ویرایش می‌کند. او می‌خواهد نامه را چاپ کند تا آن را برای اژدها ارسال کند، اما متوجه می‌شود که طول خط‌های نامه به شکلی نیست که در آن جا شود و او باید بر اساس محدودیت تعداد کاراکتر در هر خط که بر اساس سایز کاغذ نامه محاسبه می‌شود، نامه را به بخش‌های مختلفی تقسیم کند. بدین منظور از شرک می‌خواهد که برنامه‌ی دیگری برای این کار برای او بنویسد.

در این سوال باید یک رشته‌ی یک خطی را بر اساس عدد ورودی n به شکلی $align$ کنید که در هر خط تعداد ماکسیمم کاراکترها کمتر یا مساوی عدد n باشد.

نکات:

- `\n` در رشته به معنای خط جدید (`new line`) است و به عنوان **یک** کاراکتر در رشته به حساب می‌آید و جزء تعداد کاراکترهای یک خط به شمار می‌آید در صورتی که به این شکل در رشته آمده

باشد.

- اگر برای این‌که به طول مد نظر برسیم نیاز باشد که کلمه‌ای بشکند، باید به خط بعدی برود اما اسپیس‌های پشت سر هم می‌توانند شکسته شوند.
- کلمات با ` ` یا `\n` از هم جدا شده‌اند و می‌تواند هر تعدادی از آن‌ها یا ترکیبی از آن‌ها بین ۲ کلمه بیاید.

ورودی

در خط اول عدد n داده می‌شود که به معنای بیشترین طول مجاز کاراکتر در خط است و در خط بعدی یک رشته به طول m در یک خط داده می‌شود.

$$1 \leq n \leq 100$$

$$1 \leq m \leq 100100$$

خروجی

باید نسخه‌ی $align$ شده‌ی متن را چاپ کنید و در صورتی که نتوان متن را $align$ کرد، باید به جای متن نهایی عبارت `can not be aligned` را در خروجی نمایش دهید.

▼ راهنمایی

زمانی نمی‌توان متن را $align$ کرد که طول یک کلمه بیشتر از عدد ورودی باشد.

مثال‌ها

ورودی نمونه ۱

```
7
in yek matn kootah baraye test vorody ast ke tool in reshte ziad nist.
```

خروجی نمونه ۱

```
in yek
matn
kootah
baraye
test
vorody
ast ke
tool in
 reshte
 ziad
nist.
```

ورودی نمونه ۲

```
40
Exploring the \nworld of programming opens up endless possibilities. By lea
```

خروجی نمونه ۲

```
Exploring the
world of programming opens up endless
possibilities. By learning different
 languages and mastering algorithms, you
 can solve complex problems efficiently.
 Web development, in particular, offers
a creative outlet to build interactive
and dynamic websites. Understanding
concepts like hashing and digital roots
can enhance your problem-solving skills.
 Embrace the challenges and enjoy the
journey of continuous learning. Every
line of code you write brings you closer
 to becoming a proficient developer.
Keep experimenting, stay curious, and
never stop coding.
```

ورودی نمونه ۳

```
9
AsGrnxfFcd\n rJfsvPj pcbXxUP
```

خروجی نمونه ۳

```
can not be aligned
```

فایلم کو؟

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

خره پس از رمزگذاری نامه با برنامه‌ی شرک، آن را در جایی از لپتاپش ذخیره می‌کند تا روز بعد، نامه را برای چاپ ببرد. پس از شب‌به‌خیر گفتن به شرک، خره به خوابی عمیق فرو می‌رود.

روز بعد و پس از باز کردن لپتاپ خود، خره متوجه می‌شود که پس از رمزگذاری، نام فایل نیز تغییر کرده است و حالا نمی‌تواند فایل نامه را پیدا کند. او که به دلیل زود بسته شدن چاپ‌خانه‌ی مرداب می‌ترسد که نتواند نامه را در آن روز ارسال کند، شرک را با لگدی محکم از خواب بیدار می‌کند تا به کمکش بشتابد.

شرک که از لپتاپ خره سر در نمی‌آورد، تصمیم می‌گیرد نام پوشه‌های موجود را جداگانه به خره بدهد تا او یادش بیاید در کدام یک فایل را ذخیره کرده است. او به طور تصادفی فایل‌هایی را باز می‌کند و آدرس آن‌ها را در جایی می‌نویسد.

یک لیست از مسیر مطلق تمام فایل‌های کامپیوتر خره به شما داده می‌شود. وظیفه‌ی شما این است که نام تمام پوشه ($directory$) های کامپیوتر خره را به ترتیب الفبایی خروجی دهید.

هر مسیر مطلق با کاراکتر `/` آغاز می‌شود و با نام یک فایل پایان می‌یابد. نام پوشه‌های هر مسیر با کاراکتر `/` از یکدیگر جدا می‌شوند. همچنین نام هر فایل مطابق الگوی `<file_name>.<file_extension>` است.

تضمین می‌شود هیچ دو پوشه‌ای نام یکسان نداشته باشند، همچنین نام هر پوشه فقط از حروف کوچک و بزرگ الفبای انگلیسی تشکیل شده است. دقت کنید کوچکی و بزرگی حروف تاثیری در ترتیب الفبایی آن‌ها ندارد؛ اما در نام پوشه تغییر ایجاد می‌کند. برای مثال دو پوشه‌ی directory و DiRecToRy متفاوت هستند.

برای درک بیشتر صورت سوال، توضیحات نمونه را مشاهده کنید.

ورودی

در خط اول ورودی عدد n که نشان‌دهنده‌ی تعداد فایل‌ها است می‌آید.

در هر کدام از n خط بعدی، یک رشته با طول حداکثر ۱۰۰۰ کاراکتر شامل حروف کوچک و بزرگ الفبای انگلیسی و کاراکترهای `.` و `/` می‌آید که مسیر یک فایل را مشخص می‌کند.

$$1 \leq n \leq 50$$

تضمین می‌شود که نام پوشه‌ها و فایل‌ها حداکثر ۵۰ کاراکتر باشد.

خروجی

نام پوشه‌های کامپیوتر خره را در خط‌های جداگانه چاپ کنید.

مثال‌ها

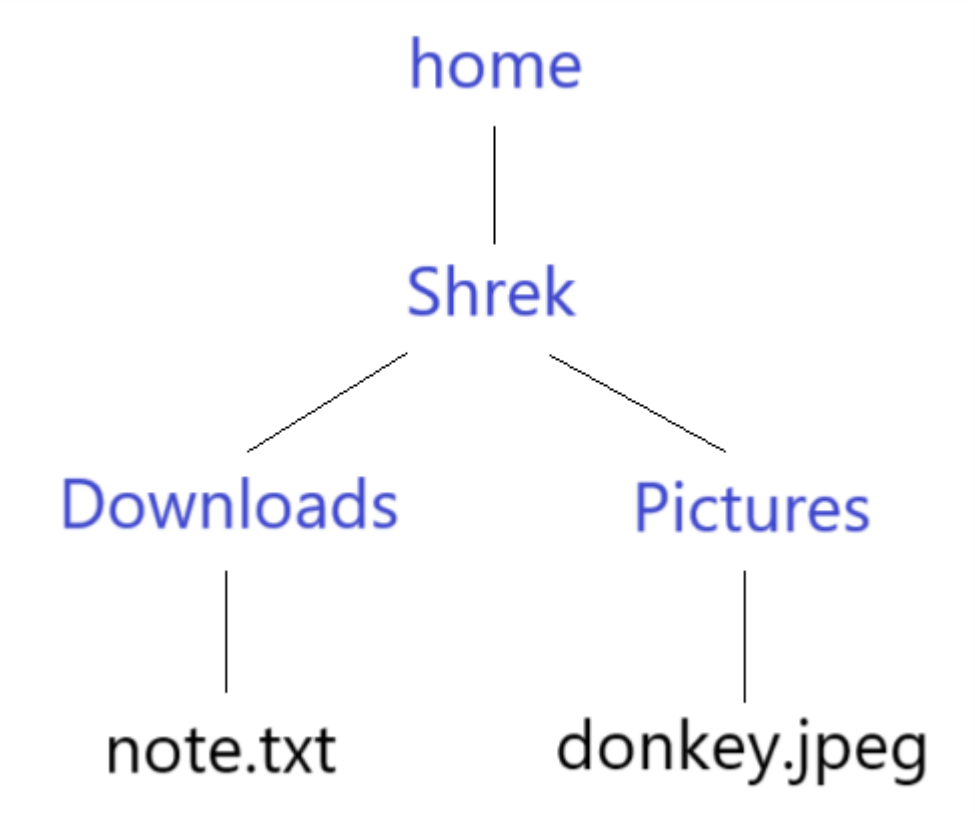
ورودی نمونه ۱

```
2
/home/Shrek/Downloads/note.txt
/home/Shrek/Pictures/donkey.jpeg
```

خروجی نمونه ۱

```
Downloads
home
Pictures
Shrek
```

ساختار فایل‌های کامپیوتر خره در این مثال به صورت زیر است. پوشه‌ها با رنگ آبی و فایل‌ها با رنگ سیاه مشخص شده‌اند.



ورودی نمونه ۲

```
4
/a.txt
/aa/a.txt
/aaa/a.txt
/aaaa/a.txt
```

خروجی نمونه ۲

```
aa
aaa
aaaa
```

ورودی نمونه ۳

```
4
/AABcaa/filename.txt
/aaABba/filename.txt
/AABABA/filename.txt
/aabbaa/filename.txt
```

خروجی نمونه ۳

```
aaABba
AABABA
aabbaa
AABcaa
```

این چیه نوشتی خره؟!

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

پس از درست کردن اعداد نامه، شرک متوجه می‌شود که خره تعداد زیادی غلط املایی و نگارشی در متنش دارد و به او توضیح می‌دهد که اگر این نامه به دست اژدها برسد، آبرویش نیست و نابود می‌شود. پس از توضیح اصول نامه‌نگاری و قالب نامه‌ها، شرک متوجه می‌شود که این روش کارساز نیست و باید از سیستمی هوشمند برای بررسی کردن اشتباهات خره روی آورد. به همین منظور شرک شروع به جست‌وجو درباره‌ی رجکس می‌کند.

رجکس یا RegEx، مخفف Regular Expressions است و برای دسته‌بندی کردن رشته‌ها طبق قالبی خاص استفاده می‌شود. برای مثال یکی از کاربردهای آن برای بررسی درست بودن قالب ایمیل وارد شده است که حتماً مطابق با name@domain.com باشد. این ابزار به این صورت عمل می‌کند که تعدادی رشته را دسته‌بندی می‌کند و تمامی آن‌ها را به نحوه‌ای خاص نشان می‌دهد. برای مثال تمامی رشته‌های زیر

```
FOP
FOOP
FOOOP
FOOOOP
FOOOOOP
...
```

به صورت

```
FO+P
```

نمایش داده می‌شوند. در ادامه‌ی سوال، توضیحات بیشتری قرار داده شده است. برای مطالعه بیشتر درباره‌ی این ابزار می‌توانید از این لینک استفاده کنید.

در این تمرین به شما یک رشته و یک رجکس داده می‌شود و باید بفهمید که آیا رشته‌ی ورودی مطابق با رجکس ورودی است یا خیر. از تمامی بخش‌های رجکس، سوال مطرح نمی‌شود و فقط کافیست که معنای کاراکترهای زیر را بلد باشید:

کاراکترها

شروع و پایان رشته

کاراکتر `^` به معنای شروع و کاراکتر `$` به معنای پایان رشته تفسیر می‌شود. یک رجکس می‌تواند با بخش‌های مختلفی از رشته تطبیق پیدا کند، اما با قرار گرفتن این دو کاراکتر در ابتدا و انتهای رجکس مشخص می‌شود که تطبیق رجکس باید از ابتدای رشته شروع شده و به انتهای آن ختم شود، یا به معنای دیگر، رجکس باید با کل رشته منطبق باشد و نمی‌تواند چیز اضافه‌ای داشته باشد.

```
aclespoiejdFOPaempoisjemd | FOP -> match
aclespoiejdFOPaempoisjemd | ^FOP$ -> no match
```

تکرار به تعداد حداقل یکی

کاراکتر + به معنای تکرار کاراکتر قبلی به تعدادی دلخواه، ولی حداقل یکی، تفسیر می‌شود.

```
FO+P = {
    FOP,
    FOOP,
    FOOOP,
    FOOOOP,
    ...
}
--------------
st+rin+g = {
    string, strinng, strinnng, strinnnng, ...
    sttring, sttrinng, sttrinnng, sttrinnnng, ...
    stttring, stttrinng, stttrinnng, stttrinnnng, ...
    sttttring, sttttrinng, sttttrinnng, sttttrinnnng, ...
    ...
}
```

تکرار به تعداد دلخواه

کاراکتر `*` به معنای تکرار کاراکتر قبلی به تعدادی دل‌خواه تقسیر می‌شود.

```
FO*P = {
    FP,
    FOP,
    FOOP,
    FOOOP,
    FOOOOP,
    ...
}
--------------
st*rin*g = {
    srig, sring, srinng, srinnng, srinnnng, ...
    strig, string, strinng, strinnng, strinnnng, ...
    sttrig, sttring, sttrinng, sttrinnng, sttrinnnng, ...
    stttrig, stttring, stttrinng, stttrinnng, stttrinnnng, ...
    sttttrig, sttttring, sttttrinng, sttttrinnng, sttttrinnnng, ...
    ...
}
```

صفر یا یک بار تکرار

کاراکتر `?` به معنی تکرار کاراکتر قبلی به تعداد صفر یا یک بار است. به عبارتی کاراکتر قبل از آن می‌تواند بیاید یا نیاید.

```
FO?P = {FP, FOP}
F?O?P? = { , F, O, P, FO, FP, OP, FOP}
```

اعداد

عبارت `d\` می‌تواند به عنوان یک رقم تفسیر شود.

```
FO\dP = {FO0P, FO1P, FO2P, FO3P, FO4P, FO5P, FO6P, FO7P, FO8P, FO9P}
\d\d\d\d\d\d\d\d\d\d = {
    09122284560,
    09122905043,
    09122339836,
    09122208297,
    09122668888,
    ...
}
```

لازم است که حواستان به escape characterها هم باشد، به این معنی که `d\\\` و `d\` معنای یکسانی ندارند.

تضمین می‌شود که تمامی رجکس‌ها با کاراکتر شروع رشته شروع شوند و با کاراکتر پایان رشته به اتمام برسند.

ورودی

در خط اول ورودی یک رشته به شما داده می شود. در خط دوم هم رجکسی که باید رشته‌ی خط اول با آن مطابقت داده شود را دریافت می‌کنید.

خروجی

اگر رشته با رجکس منطبق بود عبارت `Yes` و در غیر این صورت عبارت `No` را چاپ کنید.

مثال

ورودی نمونه

```
sp234aiin
^sp\d*ab?i+n$
```

خروجی نمونه

```
Yes
```

نامه به اژدها

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

پس از یافتن اشتباهات خره با استفاده از رجکس و بررسی کردن قالب‌ها، خره به دنبال راهی برای اصلاحشان در متن می‌گردد و دوباره از شرک کمک می‌خواهد. پس از بررسی برنامه‌هایی که در اینترنت قرار دارند، شرک متوجه می‌شود که هیچ کدام از آن‌ها یا به درد لپتاپ خره نمی‌خورند و یا رایگان نیستند. شرک که بسیار خسیس است و از نصب برنامه‌های کرک شده نیز می‌ترسد، تصمیم می‌گیرد که یک برنامه‌ی ویرایش متن بسازد و آن را روی لپتاپ خره اجرا کند.

در این سوال به شما متنی داده می‌شود و باید تعدادی عمل روی آن انجام دهید. روی این متن یک نشانگر متن (cursor) قرار دارد و به صورتی که در ادامه توضیح داده می‌شود قادر به حرکت است. هر دستوری که داده می‌شود باید با در نظر گرفتن مکان کنونی نشانگر روی متن لحاظ شود. دقت کنید که نشانگر در هر لحظه روی یکی از کاراکترها قرار دارد و نمی‌تواند بین آن‌ها قرار بگیرد.

لیست عملیات

حرکت دادن نشانگر

```
move <number>
```

با این دستور باید نشانگر را به میزان گفته شده جابه‌جا کنید. در صورتی که عدد داده شده مثبت باشد، نشانگر را به همان تعداد کاراکتر روی متن به راست جابه‌جا کنید و در صورتی که عدد وارد شده منفی باشد، نشانگر را به چپ حرکت دهید.

مکان نشانگر

```
position
```

با وارد شدن دستور بالا، باید مکان کنونی نشانگر را در یک خط جدا نمایش دهید. در نمایش مکان نشانگر باید اندیس کاراکتری که نشانگر روی آن قرار دارد را ذکر کنید. برای مثال فرض کنید نشانگر روی کاراکتر پنجم قرار دارد. این مکان را مانند زیر نمایش می‌دهیم:

```
4
```

مکان اولیه‌ی نشانگر روی اولین کاراکتر است. یعنی مکان اولیه به صورت زیر است:

```
0
```

شیفت کاراکتر

```
shift <number> <amount>
```

مکان کنونی نشانگر را X می‌نامیم. عملیات شیفت را در بازه X تا X + number به اندازه amount اجرا می‌کنیم. (عملیات روی جایگاه X اجرا می‌شود اما روی جایگاه X + number اجرا نمی‌شود.)

عملیات شیفت به این صورت است که اگر قرار باشد کاراکتری به اندازه‌ی k مقدار شیفت بخورد، باید با kمین کاراکتر بعد از خود در الفبای انگلیسی جابه‌جا شود (شیفت به صورت چرخشی صورت می‌گیرد.) برای مثال کاراکتر c بعد از ۴ شیفت به کاراکتر g تبدیل شده و کاراکتر z بعد از ۳ شیفت به c تبدیل می‌شود. اگر مقدار

`<amount>` منفی بود یعنی شیفت باید در جهت منفی انجام شده و کاراکتر به اندازه‌ی قدرمطلق `<amount>` با کاراکتر قبل از خود جابه‌جا شود؛ به این معنی که با شیفت ۴- کاراکتر a تبدیل به w می‌شود.

دقت کنید که عملیات شیفت تغییری در کوچک یا بزرگ بودن کاراکترها نمی‌دهد؛ یعنی a بعد از ۳ شیفت به d رفته و A هم بعد از ۳ شیفت به D تبدیل می‌شود.

حروف بزرگ

```
upper <number>
```

مکان کنونی نشانگر را X می‌نامیم. عملیات upper (تبدیل کاراکتر از متن به حرف بزرگ در صورت امکان) را در بازه‌ی X تا X + number اجرا می‌کنیم. (عملیات روی جایگاه X اجرا می‌شود اما روی جایگاه X + number اجرا نمی‌شود.)

حروف کوچک

```
lower <number>
```

مکان کنونی نشانگر را X می‌نامیم. عملیات lower (تبدیل کاراکتر از متن به حرف کوچک در صورت امکان) را در بازه X تا X + number اجرا می‌کنیم. (عملیات روی جایگاه X اجرا می‌شود اما روی جایگاه X + number اجرا نمی‌شود.)

توجه کنید که سه دستور بالا تنها روی کاراکترهای `a-z` و `A-Z` (حروف انگلیسی) تاثیر می‌گذارند و در غیر این صورت تاثیری روی کاراکتر اعمال نمی‌شود. مکان نشانگر نیز به تعداد $number$ جلو می‌رود. اگر تعداد کاراکترهای موجود بعد از مکان کنونی نشانگر از $number$ کمتر باشد و نشانگر پس از طی کردن $number$ کاراکتر از متن خارج شود (کاراکتری بعد از آن نباشد) دستور اجرا نمی‌شود و عبارت زیر در یک خط جدا چاپ می‌شود:

```
Not enough characters.
```

مقدار $number$ در سه دستور بالا عددی نامنفی است.

در تمامی دستورات در صورت خارج شدن نشانگر از متن (کاراکتری بعد از آن نباشد)، دستور اجرا نمی‌شود و عبارت زیر در یک خط جدا چاپ می‌شود:

```
Not enough characters.
```

شمارش

```
count <substring>
```

با وارد شدن این دستور باید تعداد دفعاتی که `<substring>` در متن ظاهر شده است را در خروجی چاپ کنید. دقت کنید که مهم نیست این زیر رشته در کدام بخش از متن شما ظاهر شده و به عنوان مثال `hre` هم در `shrek` شمرده می‌شود. این دستور نسبت به بزرگ یا کوچک بودن حروف حساس است.

دستور شمارش تغییری بر مکان نشانگر ایجاد نمی‌کند.

جابه‌جایی

```
replace <word_1> <word_2>
```

در این دستور، شما لازم است `<word_1>` های موجود در متن را با `<word_2>` جابه‌جا کنید. دقت کنید لزومی ندارد که `<word_2>` در متن ظاهر شده باشد. پس از انجام این دستور، نشانگر متن به اول متن (اندیس ۰) برده می‌شود.

ترجمه به زبان هندی

در این دستور باید متن را به هندی ترجمه کنید.

```
translate Hindi
```

در ترجمه به زبان هندی، شما باید تمامی کاراکترهای `t` را با کاراکتر `d` و تمامی کاراکترهای `T` را با کاراکتر `D` جابه‌جا کنید. این دستور تغییری روی مکان نشانگر و کاراکترهایی غیر از `t` و `T` اعمال نمی‌کند.

ترجمه به زبان فرانسوی

در این دستور هم ترجمه به زبان فرانسوی لازم است.

```
translate French
```

در اینجا شما باید تمامی کاراکترهای `r` را با کاراکتر `q` و تمامی کاراکترهای `R` را با کاراکتر `Q` جابه‌جا کنید. این دستور تغییری روی مکان نشانگر و کاراکترهایی غیر از `r` و `R` ایجاد نمی‌کند.

ورودی

در خط اول ورودی، متن اولیه به شما داده می‌شود. در خط‌های بعدی هم دستورات به ترتیب در ورودی نوشته می‌شوند تا در نهایت به `&&` برسیم و برنامه به پایان برسد.

تضمین می‌شود که طول متن پس از اعمال هیچ عملی از ۱۰۰۰۰ کاراکتر طولانی‌تر نشود.

در دستورات ورودی وجود بیش از یک اسپیس بین اجزای دستور در کارکرد آن تاثیری ندارد. (این مورد را در دستورات count و replace در نمونه دوم می‌توانید ببینید.)

خروجی

در حین اجرای برنامه، خروجی‌هایی که در بعضی دستورات ذکر شده است را چاپ کنید. در انتهای برنامه هم متن نهایی را در خروجی چاپ کنید.

مثال

ورودی نمونه ۱

```
don't tell me what to do
translate Hindi
&&
```

خروجی نمونه ۱

```
don'd dell me whad do do
```

ورودی نمونه ۲

```
She was a ship of the old school, rather small if anything; with an old-fas
translate French
move 30
position
move 7
shift 5 10
position
translate Hindi
upper 8
lower 5
shift 10 15
translate Hindi
count ld
replace  ng  idjs
&&
```

خروجی نمونه ۲

```
30
42
3
She was a ship of dhe old school, qadroa cMALL IF anydhxcv; lxsw an old-fas
```