Grammar

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

برای یک گرامر، یک متود پیاده سازی کنید که به عنوان ورودی، یک شیء گرامر و یک رشته را دریافت کند و درصورتی که رشته توسط گرامر تولید شود، `Accepted` و درغیراین صورت، `Rejected` را به عنوان خروجی برگرداند.

در گرامر ورودی، متغیرها داخل `<>` هستند، `#` نشان‌دهنده‌ی nullable transition است و قواعد مختلف با `|` از همدیگر جدا می‌شوند.

ورودی

نحوه ورودی گرفتن برنامه به این صورت است که در ابتدا یک عدد n که بیانگر تعداد متغیرهای گرامر می‌باشد، در خط اول می‌آید. سپس، در هریک از n خط بعدی، یکی از قواعد گرامر به عنوان ورودی به شما داده می‌شود. در انتها به عنوان آخرین خط ورودی، یک رشته داده می شود که باید پذیرفته شدن یا نشدن آن را توسط گرامر بررسی کنید.

توجه: به فاصله‌ها در ورودی دقت کنید که دقیقا ورودی را به همین فرمت باید دریافت کنید.

$$1 \leq n \leq 10$$

خروجی

خروجی برنامه‌ی تنها شامل یک خط است که باید پذیرفته شدن یا نشدن رشته مربوطه را در گرامر به ترتیب با `Accepted` یا `Rejected` نمایش دهید.

مثال

ورودی نمونه ۱

```
3
<S> -> a<S>b | a<A> | b<B>
<A> -> a<A> | #
<B> -> b<B> | #
aaab
```

خروجی نمونه ۱

```
Accepted
```

ورودی نمونه ۲

```
4
<S> -> <S>a | <S>b | <A>a | <B>b
<A> -> ab<A> | <B>ca | #
<B> -> b<B> | <C>f
<C> -> a<C> | #
abbfcaba
```

خروجی نمونه ۲

```
Rejected
```

PDA Calculator

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۶۴ مگابایت

با استفاده از PDA یک ماشین حساب بسازید که به عنوان ورودی یک عبارت ریاضی را دریافت کند و درصورت معتبر بودن، مقدار محاسبه‌شده آن را نمایش دهد و درصورتی که ورودی معتبر نباشد، پیام `INVALID` را چاپ کند.

ماشین حساب شما باید دارای عملگر های زیر باشد:

۱. پرانتز: $($، $)$

۲. چهار عمل اصلی: $+$، $-$، $*$، $/$

۳. توان: $^{\wedge}$

۴. جذر: $sqrt$

۵. توابع مثلثاتی: sin, cos, tan

۶. تابع قدرمطلق: abs

۷. توابع نمایی: exp, ln

نکات مهم:

- ورودی تنها شامل یک خط است که در آن یک رشته به شما داده می‌شود.
- ورودی عملگرهای 4 تا 7 داخل پرانتز قرار خواهند گرفت.
- در رشته ورودی، متغیر وجود نخواهد داشت و ورودی تنها شامل اعداد حقیقی خواهد بود.
- هرگونه ورودی خارج از دامنه‌ی تابع و تقسیم بر 0 نیز باعث `INVALID` شدن می شود.
- حاصل عبارت خروجی باید به صورت اعشاری و با دو رقم اعشار باشد.

ورودی

ورودی برنامه شامل یک خط است که یک رشته از اعداد، عملگرها و پرانتزها می‌باشد.

خروجی

اگر ورودی معتبر باشد، در یک خط حاصل آن باید نمایش داده شود. درغیراین‌صورت، باید عبارت `INVALID` چاپ شود.

مثال

ورودی نمونه ۱

```
10 / 2 + 4 * -3.5
```

خروجی نمونه ۱

```
-9.00
```

ورودی نمونه ۲

```
(10 + 9) * 4)
```

خروجی نمونه ۲

```
INVALID
```

ورودی نمونه ۳

```
sin(ln(5 ^ 6 * 3 - cos(sin(6))))
```

خروجی نمونه ۳

```
-0.97
```

Turing Machine

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۱۲۸ مگابایت

برنامه‌ای بنویسید که رشته‌ی کدگذاری شده‌ی یک ماشین تورینگ را دریافت کرده، آن را decode کند و سپس با توجه به آن، ماشین تورینگ مربوطه را پیاده‌سازی کند و هریک از رشته‌های ورودی را بر روی ماشین تست کند. (برای اطلاع از نحوه کدگذاری به اسلایدهای درس مراجعه کنید.)

ورودی

اولین خط از ورودی یک رشته باینری (متشکل از 0 و 1) است که رشته‌ی کدگذاری شده‌ی یک ماشین تورینگ است. در خط دوم ورودی، عدد صحیح n می‌آید که بیانگر تعداد رشته‌هایی است که باید روی ماشین به عنوان ورودی تست شود. سپس در هریک از n خط بعدی، در هرخط یک رشته کدگذاری‌شده به عنوان ورودی می‌آید که باید روی ماشین پیاده‌سازی‌شده تست شوند.

$$1 \leq n \leq 10$$

خروجی

خروجی برنامه دقیقا شامل n خط است که در هرخط، نتیجه تست هریک از ورودی‌ها روی ماشین تورینگ، به ترتیب چاپ می‌شوند. درصورت قبولی هر ورودی، باید عبارت `Accepted` و درغیراین‌صورت، عبارت `Rejected` چاپ شود.

نکات مهم

- کد حالت آغازین را ۱ درنظر بگیرید.
- کد حالت پایانی را $1^{(numberofstates)}$ درنظر بگیرید.

- کد کاراکتر خالی (blank) را در نوار ۱ درنظر بگیرید.
- تضمین می شود در ماشین تورینگ داده شده، حلقه بی نهایت رخ نمی دهد.

مثال

ورودی نمونه ۱

```
101101011011001010110101
3
11011011
110111011
```

خروجی نمونه ۱

```
Accepted
Rejected
```

ورودی نمونه ۲

```
10111011011111011001011011111011011001101101101101100110111011011101101100110111111011101101
5
111011111011111111
1111101110111
1110111011111011111
111011111
```

خروجی نمونه ۲

```
Rejected
Rejected
Accepted
Accepted
```