شهر ممد

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- سطح:ساده
- طراح: نیما سلطانی

ممد که به تازگی شهردار تهران شده تصمیم گرفته به مشکل آلودگی هوا پایان بده.برای این کار همه ساختمون های شهر رو خراب کرده و تعداد زیادی ساختمون در یک خط ساخته که ارتفاعشون فرق داره.اون معتقده با این کار باد بیشتری به نقاط شهر میرسه و دیگه از آلودگی هوا خبری نخواهد بود.حالا که کار ساخت و ساز تموم شده،ممد میخواد یکی از واحد ها رو برای خودش برداره.برای این کار اول بلندترین توالی صعودی از ساختمون هارو انتخاب میکنه بعد در اون توالی ساختمونی رو انتخاب میکنه که مجموع اندازه اختلاف ارتفاعش با دو ساختمون مجاور بیشینه باشه.دقت کنید ساختمون انتخابی حتما باید درون توالی باشه و ممد در ابتدا یا انتهای بازه ساکن نمیشه

ورودی

ورودی تنها شامل دو خط است که در خط اول تعداد ساختمان ها و در خط دوم ارتفاع آنها با فاصله از هم آمده است.

خروجی

در خط اول توالی مورد انتخاب ممد و در خط بعدی ارتفاع ساختمون انتخابی ممد رو چاپ کنید

مثال

ورودی نمونه ۱

```
9
1 2 1 2 3 8 10 11 9
```

خروجی نمونه ۱

```
1 2 3 8 10 11
8
```

رمزعبور قوی

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- سطح: متوسط
- طراح: زهرا عزیزی

گروهی از دوستان شما تصمیم به راه اندازی یک وبسایت برای مدیریت وضعیت رزرو کمد های دانشکده گرفته اند تا بتوانند وضعیت هر کمد را به خوبی مدیریت کنند و گزارش آن ها را به دست مسئولین مربوطه برسانند. آن ها برای افزایش امینت هر کاربر، از شما به عنوان یک برنامه نویس مطمئن خواسته اند تا برنامه ای بنویسید که رمزعبوری که هر کاربر هنگام ثبت نام انتخاب می کند را بررسی کند و سطح قدرت رمز انتخاب شده را به کاربر نمایش دهد. سطح قدرت هر رمزعبور به صورت زیر تعریف می شود:

سطوح قدرت رمزعبور

Very Strong

- **معیارها:**
  - حداقل ۱۲ کاراکتر طول داشته باشد.
  - حداقل شامل یک حرف بزرگ باشد.
  - حداقل شامل یک حرف کوچک باشد.
  - حداقل شامل یک رقم باشد.
  - حداقل شامل یک کاراکتر خاص `()*&^%$#@!` باشد.
- **مثال:** `P@ssw0rd1234!`

Strong

- **معیارها:**
  - حداقل ۸ کاراکتر طول داشته باشد.
  - حداقل شامل یک حرف بزرگ باشد.

- حداقل شامل یک حرف کوچک باشد.
- حداقل شامل یک رقم باشد.

- **مثال:** `StrongPass123`

Medium

- **معیارها:**
  - حداقل ۶ کاراکتر طول داشته باشد.
  - حداقل شامل یک حرف (بزرگ یا کوچک) باشد.
  - حداقل شامل یک رقم باشد.
- **مثال:** `Medi12`

Weak

- **معیارها:**
  - حداقل ۴ کاراکتر طول داشته باشد.
  - فقط شامل حروف (بزرگ یا کوچک) باشد.
  - شامل هیچ رقمی یا کاراکتر خاصی نباشد.
- **مثال:** `WeakPass`

Very Weak

- **معیارها:**
  - کمتر از ۴ کاراکتر طول داشته باشد.
  - فقط شامل اعداد یا فقط شامل حروف باشد (بدون ترکیب).
- **مثال:** `abc` یا `123`

توجه داشته باشید برنامه شما باید با استفاده از رجکس، سطح قدرت هر رمز را بررسی کند. استفاده از راه های دیگر مورد پذیرش نیست.

ورودی

ورودی تنها شامل یک خط است که در آن رمزعبور s آمده است.

$$1 \leq |s| \leq 20$$

خروجی

در تنها خط خروجی، سطح قدرت رمزعبور کاربر را چاپ کنید. به **بزرگی و کوچکی حروف** توجه داشته باشید.

ورودی نمونه ۱

```
4
```

خروجی نمونه ۱

```
Very Weak
```

ورودی نمونه ۲

```
ZYIx9lhrWwT
```

خروجی نمونه ۲

```
Strong
```

پیام محرمانه

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- سطح: متوسط
- طراح: روزبه سلطانی

یک سازمان مخفی به نام "سایه‌ها" اطلاعات فوق‌سری را بین اعضای خود جابه‌جا می‌کند. اما برای جلوگیری از لو رفتن اطلاعات، تمامی پیام‌ها با یک روش رمزنگاری خاص کدگذاری می‌شوند.

این روش رمزنگاری، که توسط یکی از بهترین متخصصان امنیت سایبری طراحی شده، بر اساس ساختار حروف در کلمات و موقعیت آن‌ها در جمله کار می‌کند. به این ترتیب، تنها کسانی که از الگوریتم دقیق آن اطلاع داشته باشند، قادر به رمزگشایی پیام‌ها خواهند بود.

امشب، یکی از این پیام‌های محرمانه به دست شما رسیده است. مأموریت شما ساده است: این پیام را برای ارسال ایمن، رمزگذاری کنید.

برای رمزگذاری متن، ابتدا جمله به **کلمات جداگانه** تقسیم می‌شود و سپس **هر کلمه به‌صورت مستقل رمزگذاری می‌شود**. تغییر مقدار `ASCII` حروف بر اساس **موقعیت کلمه در جمله** و **بزرگ‌تر یا کوچک‌تر بودن حروف نسبت به حرف قبل از خودشان** انجام می‌شود.

قوانین رمزنگاری:

۱. **حروف در هر کلمه به‌صورت صعودی و نزولی تغییر میکنند:**

- اگر یک حرف از حرف قبل خودش بزرگ‌تر یا مساوی آن باشد، مقدار `ASCII` آن افزایش می یابد.
- اگر یک حرف از حرف قبل خودش کوچک‌تر باشد، مقدار `ASCII` آن کاهش می یابد.
- حرف اول هر کلمه همیشه افزایش می‌یابد.
- توجه داشته باشید که مقایسه بزرگ‌تر یا کوچک‌تر بودن حروف بر اساس مقدار `ASCII` آن ها است.
- کاراکتر فاصله ( ) تغییری نمی‌کند و باید در همان موقعیت خود در جمله حفظ شود.

۲. **مقدار تغییر `ASCII` برابر با شماره‌ی کلمه در جمله است:**
   - کلمه‌ی اول تغییر ۱ واحدی دارد.
   - کلمه‌ی دوم تغییر ۲ واحدی دارد.
   - کلمه‌ی سوم تغییر ۳ واحدی دارد.
   - و الی آخر...

نکته :

۱. اگر مقدار `ASCII` از `z` عبور کند، باید دوباره از `a` شروع شود.
۲. اگر مقدار `ASCII` از `Z` عبور کند، باید دوباره از `A` شروع شود.
۳. اگر مقدار `ASCII` کمتر از `a` شود، باید دوباره از `z` ادامه پیدا کند.
۴. اگر مقدار `ASCII` کمتر از `A` شود، باید دوباره از `Z` ادامه پیدا کند.

نمونه‌ی فرآیند رمزگذاری روی یک کلمه:

مثال برای کلمه‌ی `"come"` که در جایگاه اولین کلمه در جمله است.

مرحله‌ی ۱: اعمال تغییر ۱ واحدی بر روی حرف اول → `c → d`

مرحله‌ی ۲: `o` بزرگ‌تر از `c` است → افزایش ۱ واحدی → `o → p`

مرحله‌ی ۳: `m` کوچک‌تر از `o` است → کاهش ۱ واحدی → `m → l`

مرحله‌ی ۴: `e` کوچک‌تر از `m` است → کاهش ۱ واحدی → `e → d`

نتیجه: `"come" → "dpld"`

ورودی

یک جمله‌ی متشکل از چندین کلمه که با فاصله از هم جدا شده‌اند. تمامی کاراکترهای جمله **حروف انگلیسی کوچک و بزرگ** هستند.

خروجی

برنامه باید **نسخه‌ی رمزگذاری‌شده‌ی جمله را چاپ کند**.

مثال

ورودی نمونه ۱

```
this is a secret message
```

خروجی نمونه ۱

```
ugjt ku d wayvax rzxxvlz
```

ورودی نمونه ۲

```
come over here
```

خروجی نمونه ۲

```
dpld qxct kbub
```

راز مخفی جیمیل

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- سطح سوال: متوسط
- طراح: احسان حبیب آگهی

چند دانشجوی فارغ التحصیل مهندسی کامپیوتر که دوستان هم بودند یک **سایت فروشگاهی آنلاین** را به‌عنوان پروژه‌ی خود راه‌اندازی کردند. همه چیز به‌خوبی پیش می‌رفت تا اینکه متوجه شدند یک کاربر مشکوک، بیش از حد مجاز از **تخفیف‌های ویژه** استفاده کرده است. بررسی پایگاه داده نشان داد که این فرد از ایمیل‌های مختلفی برای ثبت‌نام استفاده کرده، اما الگویی عجیب در آن‌ها وجود داشت.

پس از کمی بررسی معلوم شد یک کاربر کنجکاو با ایجاد چندین حساب کاربری که در ظاهر متفاوت بودند، اما در واقع به یک ایمیل اصلی ختم می‌شدند، سیستم را دور زده بود. نمونه‌ای از ایمیل‌های او به این شکل بود:

ایمیل های مشابه:

```
ehsan.sale@gmail.com

eh.san.sale@gmail.com

ehsan.sale+vip@gmail.com

ehsan.sale+discount@gmail.com
```

ایمیل اصلی:

```
ehsansale@gmail.com
```

جیمیل نقطه‌ها را در نام کاربری نادیده می‌گیرد و هر چیزی که بعد از + بیاید، اثری در آدرس واقعی ندارد. بنابراین، تمام این ایمیل‌ها در واقع یکی بودند.

در این مورد میتونین اینجا بیشتر بخونین 🙃🧸

برای جلوگیری از این تقلب، دانشجویان تصمیم گرفتند برنامه‌ای بنویسند که ایمیل‌های تکراری را شناسایی کند. برنامه باید:

۱. **تمام نقطه‌های موجود در نام کاربری قبل از `@` را حذف کند.**
۲. **هر چیزی که بعد از `+` می‌آید را نادیده بگیرد.**
۳. **ایمیل‌ها را به ساده‌ترین حالتشان تبدیل و چاپ کند. اگر یک ایمیل نامعتبر بود عبارت `NO` را چاپ کند**

نکته ۱ یک آدرس ایمیل معتبر در دامنه خود (بعد از نقطه) حداقل دو حرف دارد

نکته ۲ یک جیمیل حداکثر یک `+` میتواند داشته باشد

پس از اجرای برنامه، مشخص شد که این کاربر چندین حساب با ایمیل یکسان ساخته بود. تیم برنامه‌نویسی با اعمال این فیلترها در سیستم، مشکل را برطرف کرد و فروشگاه آنلاین دیگر در برابر این ترفند زیرکانه 🔮 مقاوم شد.

ورودی نمونه ۱

```
invalid-email
```

خروجی نمونه ۱

```
NO
```

ورودی نمونه ۲

```
ehsan.2+fiends@gmail.com
```

خروجی نمونه ۲

```
ehsan2@gmail.com
```

خاطرات گم شده

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- سطح سوال: سخت
- طراح: امیررضا یزدان پناه

در شرکت *Lumon*، خاطرات کارمندان به وسیله‌ی فرآیندی مرموز به نام ***Severance*** بین زندگی کاری و شخصی تقسیم می‌شود. برای بازنشانی خاطرات کاری، یک الگوریتم سرّی، رشته‌ی ذخیره‌شده‌ی حافظه را به شیوه‌ای عجیب تغییر مکان می‌دهد. به عنوان یک تکنسین در *Lumon*، وظیفه‌ی شما بازیابی شکل نهایی خاطرات کاری یک کارمند پس از چندین دور اجرای این الگوریتم است.

شما یک رشته‌ی حافظه `S` به طول `N` دارید که از حروف کوچک انگلیسی تشکیل شده است. برای به هم ریختن حافظه، شرکت *Lumon* از یک آرایه‌ی عدد صحیح `A` به طول `N` استفاده می‌کند. این الگوریتم، یک جایگشت روی حروف `0` تا `N - 1` را به صورت زیر تعریف می‌کند:

برای هر اندیس `i` (از 0 شروع می شوند):

$$P(i) = (i + A[i]) mod N$$

یک مرحله تبدیل، رشته‌ی جدیدی `T` ایجاد می‌کند به طوری که:

$$T[i] = S[P(i)]$$

نکته: تمامی حروف به صورت همزمان آپدیت می شوند.

این فرآیند دقیقا `K` بار اعمال می‌شود. وظیفه‌ی شما محاسبه‌ی رشته‌ی نهایی حافظه پس از اجرای دقیقاً `K` تبدیل است.

ورودی

خط اول ورودی شامل دو عدد صحیح `N` و `K` است (به ترتیب):

$$1 \leq N \leq 10^5$$

$$1 \leq K \leq 10^9$$

خط دوم شامل رشته‌ی `S` به طول `N` است (فقط شامل حروف کوچک انگلیسی).

$$|S| = N$$

خط سوم شامل `N` عدد صحیح از `A[0]` تا `A[N - 1]` است که با فاصله از هم جدا شده اند.

$$1 \leq A[i] \leq N - 1$$

تضمین می شود که نگاشت

$$i \to (i + A[i]) mod N$$

یک جایگشت از مجموعه‌ی {N-1,...,1,0} تشکیل می‌دهد.

خروجی

رشته‌ی نهایی حافظه را پس از اعمال دقیقاً K بار تبدیل چاپ کنید.

ورودی نمونه ۱

```
5 2
abcde
1 2 3 4 0
```

خروجی نمونه ۱

```
dcbae
```

توضیح نمومه 1:

ابتدا جایگشت P را محاسبه می کنیم:

۱. $P(0) = (0 + 1) mod 5 = 1$

۲. $P(1) = (1 + 2) mod 5 = 3$

۳. $P(2) = (2 + 3) mod 5 = 0$

۴. $P(3) = (3 + 4) mod 5 = 2$

۵. $P(4) = (4 + 0) mod 5 = 4$

تبدیل اول:

رشته‌ی جدید به صورت زیر خواهد بود:

۱. $T[0] = S[1] = b$

۲. $T[1] = S[3] = d$

۳. $T[2] = S[0] = a$

۴. $T[3] = S[2] = c$

۵. $T[4] = S[4] = e$

بنابراین، پس از یک گام، S به رشته‌ی "bdace" تبدیل می‌شود.

تبدیل دوم:

اعمال همان جایگشت بر روی "bdace" منجر به رشته‌ی "dcbae" می‌شود.

پازل (امتیازی)

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- طراح: مهرسا سمیع‌زاده

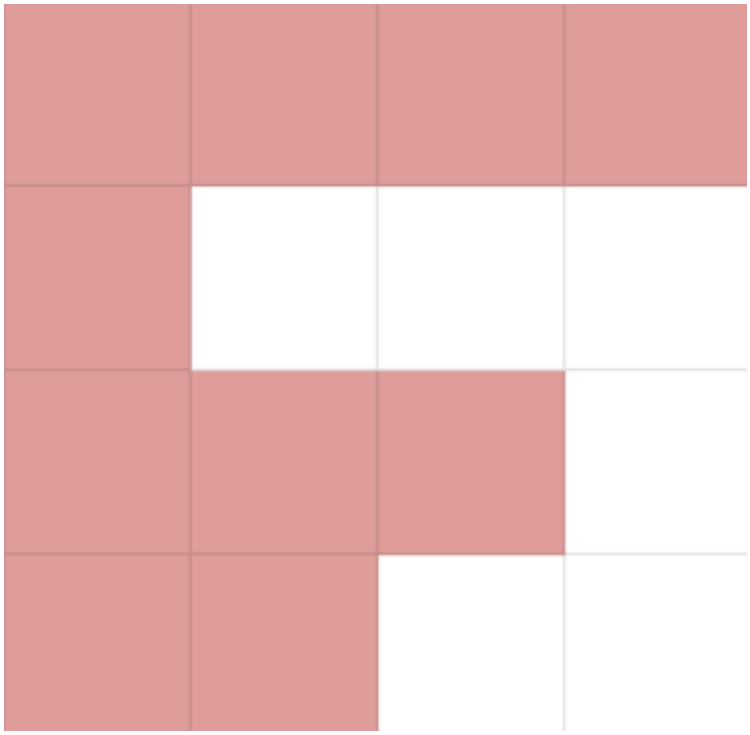
تکه های پازل های ما گم شده و نیاز به برنامه‌ای داریم تا بتواند تشخیص بدهد کدام سه قطعه میتوانند پازل را کامل کنند.

صفحات ما از grid های $4 * 4$ تشکیل شده‌اند و برای تشکیل هر پازل کامل به سه قطعه نیازداریم تا با در کنار هم چیدن انها صفحه کامل را تشکیل دهیم. هر صفحه را با آرایه یک بعدی متشکل از 4 عدد دسیمال، نشان میدهیم، که فرم باینری چهار بیتی عنصر هر خانه، نمایانگر یک ردیف از صفحه میباشد. که 0 به معنای خانه خالی و 1 به معنای خانه پر است.

با گرفتن سه صفحه ورودی، سعی کنید حالتی از آنهارا پیدا کنید که با **کمترین** میزان تغییر(چرخش صفحات)، بتوانند پازل را تکمیل کنند.

توجه کنید که صرفا از آرایه‌های **یک بعدی** می‌توانید استفاده کنید.

برای مثال آرایه `arr1 = [3,7,1,15]` را میتوان به شکل زیر تصور کرد:



```
3 = 0 0 1 1
7 = 0 1 1 1
1 = 0 0 0 1
15 = 1 1 1 1
```

هر ردیف متناظر با یک عنصر میباشد.

توجه کنید که تنها مجاز به چرخش (90 درجه) چپ یا راست صفحات هستید

ورودی

سه آرایه چهار عضوی در سه خط متوالی داده می‌شود. به طوری که هر عضو آن:

$$0 \leq X \leq 15$$

خروجی

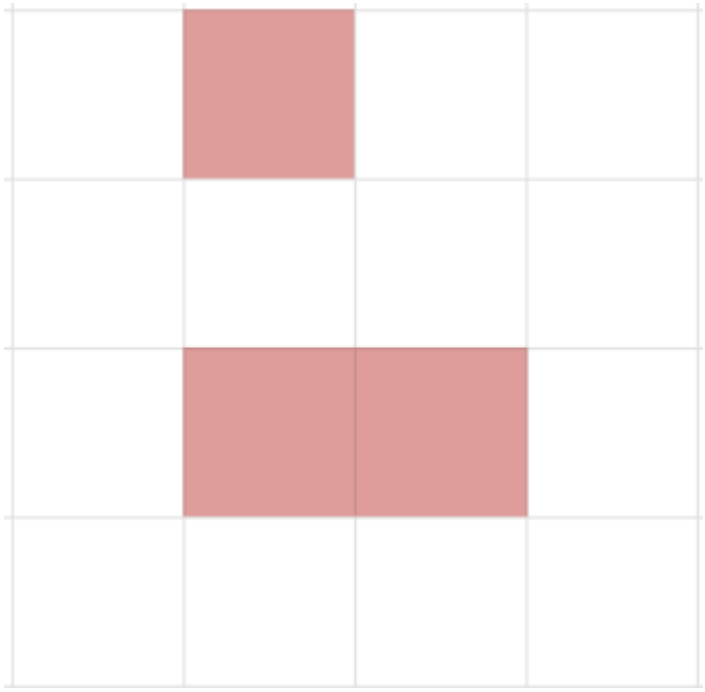
در صورتی که سه قطعه داده شده، میتوانستند پازل را کامل کنند، `Complete Set` و فرم نهایی سه آرایه داده شده را پرینت کنید. درصورت تداخل خانه های پر در *همه‌ی حالات* `not matching` پرینت شده و در

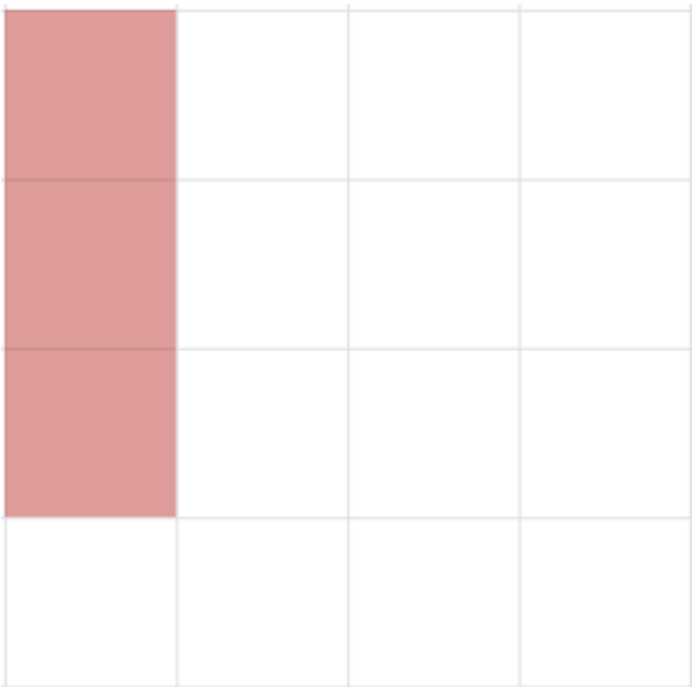
شرایطی که اورلپ وجود ندارد، صفحه‌ی چهارمی که پازل را کامل می‌کند به همراه عبارت `Missing Piece`پرینت شود.

ورودی نمونه 1

```
0 1 1 1
3 7 1 15
0 6 0 2
```

آرایه اول را نشان دادیم،، دو آرایه دیگر به شکل زیر نمایش داده میشوند:





خروجی نمونه 1

```
Complete Set
[0, 1, 1, 1]
[15, 8, 14, 12]
[0, 6, 0, 2]
```

برره کامپایلر ۱.۱ (امتیازی)

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت
- طراحان : برنا ماهرانی و آریا شاکو

در برره چند هفته ای می شود که شیر فرهاد زبان برنامه نویسی بومی ساز(!) برره ۱.۱ را ابداع کرده است. اما شیر فرهاد از ابداع زبان برنامه نویسی فقط قواعد آن را بلد هست و هیچی از پیاده سازی کامپایلر آن نمیداند. قرار است شما به شیر فرهاد کمک کنید که برره کامپایلر ۱.۱ را به بازار عرضه کند!

Barare 1.1 Syntax

این زبان مخلوطی از زبان های مختلف مرسوم می باشد که کار را برای برنامه نویسان برره خیلی آسان تر کرده است. در این زبان فقط اجزا زیر وجود دارد.

if : if (conditions) { then ... }

- در اینجا condition شرطی می باشد که در آن فقط از علائم == , != , =>, <=, =! استفاده می شود.همیشه سمت چپ مقایسه متغیر و سمت راست فقط عدد می باشد.

else : else { then ... }

while : while (conditions) { then ... }

variable naming:

- متغیر ها در برره ۱.۱ مانند پایتون تعریف می شوند. بعد علامت مساوی رشته، عدد و فلوت میتواند قرار گیرد. اسم متغیر ها با عدد نمی تواند شروع بشود. با حرف بزرگ نمی تواند شروع بشود. کلمات رزرو شده بالا نمیتوانند باشند. در بین آن اسپیس نمیتوان به کار برد. در غیر این موارد هر اسمی آزاد میباشد. در آخر مقدار دهی متغیر ; فراموش نشود!

Function calling: Call functionName(var1,var2, ...);

Function defenition: functionName(var1,var2,...):

- اسم توابع حتما با حروف بزرگ شروع می شود و فاقد ارقام می باشد. محتوای تابع حتما با tab جدا می شود.
- پس از if else while حتما باید حداقل یک tab زده شود تضمین میشود برای متغیر ها لازم نیست tab ای زده شود.

ورودی

در خط اول ورودی یک عدد طبیعی n که نشان دهنده تعداد خطوط کد می باشد و در n خط بعد کد ها وارد می شوند.

$$1 \leq n \leq 100$$

نکته: خطای منطقی مد نظر این سوال نمی باشد. تضمین می شود متغیر های به کار رفته در شروط، قبلا تعریف شده اند. تضمین می شود هر تابعی که فراخوانی می شود قبلا تعریف شده است. برای شناسایی خطا های گفته شده در بخش بالا حتما از Regex استفاده شود.

خروجی

اگر کد داده شده در ورودی خطایی نداشت در خروجی عبارت du barare چاپ شود. در غیر این صورت شماره خط های ارور دار را چاپ کند.

مثال

ورودی نمونه ۱

```
7
Main():
    x = 6;
    while(x >= 7)
        x = 5;
Dubarare(st):
    while( x == 7)
        x = 6;
```

خروجی نمونه ۱

```
du barare
```

ورودی نمونه ۲

```
5
main():
    borna = 3
    aria = 3
    if ( aria == 3)
        borna = 4;
```

خروجی نمونه ۲

```
1
2
3
```

ورودی نمونه 3

```
8
main():
x = 5;
    while( x >= 8)
        Call Print(x);
y = 6.2;
while( y == 5)
        Call Sum(x,y);
z= 6;
```

خروجی نمونه 3

```
1
```