سانسورچی

ایلیا قراره به عنوان سانسورچی توی کوئرا فعالیت کنه و باز گیر کرده، مسئولین کوئرا بخاطر محدودیت حافظه روی سرورهاشون، گفتن که نمیتونن برای هر متن یه حافظه ی جدای موقت به ایلیا بدن و ایلیا مجبوره سانسوراش رو روی خود متن در همون لحظه انجام بده!

ورودی

در خط اول تعداد کلمات ممنوعه به شما داده میشه. در خط دوم کلمه(های) ممنوعه که با , جدا شدن بهتون داده میشه. در خط سوم یک متن به شما داده میشه که باید کلمات ممنوعه ازش حذف شه و به جاش، به تعدادِ دو برابرِ حروفش علامت ستاره چاپ کنید. مثلا برای کلمه ی iliya باید 10تا ستاره چاپ شه.

خروجی

در خروجی ، متن سانسور شده چاپ می شود.

ورودی نمونه ۱

```
1
spam
in yek math haavi kalameye spam ast, ma az spam bizaarim;
```

خروجی نمونه ۱

```
in yek math haavi kalameye ******** ast, ma az ******** bizaarim;
```

نکته: مجاز به استفاده از رشته ی کمکی نیستید و باید روی رشته ی اصلی که ورودی روی آن ذخیره می شود، تغییرات رو اعمال کنید.

چالش های جدید برای ایلیا

هفته ی پیش مشکل اعداد کسری رو حل کردید. حالا این هفته به یه چالش جدید داریم!! ایلیا دیتاهایی که میخواد ذخیره کنه فقط اعداد کسری نیست و اون نیاز داره که اعداد صحیح، کاراکتر و عدد اعشاری (float) رو اعداد کسری (تمرین قبل) رو ذخیره کنه و بخاطر محدودیت های زمانی و حافظه، نیاز شدید داره که از لینکدلیست استفاده کنه! یعنی هرموقع خواست توی لینک لیستش هرکدوم از دیتاهای مورد نیازش رو ذخیره کنه ولی خب تا الان با هر لینکدلیستی که کار کرده همه ی دیتاها از یک نوع بودن!! کمک کنید که بتونه این کار رو انجام بده.

نکته: نیاز به اضافه کردن فانکشن های تمرین قبل نیست و جهت خلوت بودن کدتون فقط استراکت اعداد کسری رو استفاده کنید.

نکته: نیازی به پیاده سازی توابع برای لینکدلیست نیست و میتوانید به صورت دستی دیتاتایپ هارا وارد کنید.

ورودی

درین سوال ورودی و خروجی نداریم و صرفا پیاده سازی درست لینکدلیست مد نظر است. در تابع main یک لینکدلیست با حضور تمام دیتاتایپ های گفته شده بسازید.

آرگومان ها و کار با فایل

ورودی

برنامه شما توسط یک فایل اسکریپت که شامل n خط آرگومان های تابع main است n بار اجرا میشود. برنامه هنگام اجرا شدن برای i امین بار، باید عدد i ام یک فایل file.txt را بخواند. (راهنمایی: میتوانید برای نگه داشتن i خودتان یک فایل دیگر ایجاد کنید و از آن کمک بگیرید)

$$1 <= i <= n$$

خروجی

خروجی برنامه بایستی فایلی با نام شماره دانشجویی شما باشد که شامل n خروجی به فرمت نمایش داده شده در مثال است. در خط اولِ هر یک از n خروجی تعداد رشته های موجود در آرگومان های ورودی(m=) و در m خط بعدی طول هر رشته و در خط آخر جمع ک.م.م اعداد موجود در آرگومان ها با عدد i ام فایل file.txt نمایش داده شود. اگر کمتر از ۲ عدد در آرگومانها موجود بود عبارت "none" بجای ک.م.م نوشته شود.

مثال

در این مثال کد شما با اسکریپت زیر اجرا میشود. همانطور که میبینید ۵ دستور اجرا در فایل file.sh قرار دارد پس کد شما ۵ بار پشت سر هم اجرا میشود. (n=5)

file.sh

```
./main hello guys 99
./main glad "20 smart students" are here 67 4 23 1
./main 1 and 7 and 2 and '9 '
./main make up 'one of the most beautiful numbers' which is 1729
./main 12407 6 108 23 10 are also beautiful
```

فایل file.txt باید هربار توسط کد شما باز شده، اعداد موجود در آن خوانده شود و سپس بسته شود. در این فایل n=5 تا عدد نوشته شده است.

file.txt

```
18
4
3
19
2
```

در فایل studentNo.txt که خروجی کد است، در دومین بار اجرای کد، (i=2) ابتدا تعداد رشته ها (m=4) و سپس در ۴ خط بعدی طول هررشته به ترتیب نوشته شده است. ک.م.م اعداد ۶۷، ۴، ۲۳ و ۱ عدد ۶۱۶۴ است که با عدد i=۲ ام فایل file.txt که برابر ۴ است جمع شده و ۶۱۶۴+ ۴ =۶۱۶۸ خروجی خط آخر i=2 امین اجرای کد است.

خروجی نمونه ۱

studentNo.txt

```
2
5
4
none
----------------------
4
4
17
3
4
6168
----------------------
4
3
3
3
2
```

```
17
----------------------
5
4
2
33
5
2
none
----------------------
3
3
4
9
154094942
----------------------
```

لینک لیست کامل

در این سوال قراره یک لیست پیوندی کامل یک طرفه را پیاده سازی کنیم. هر Node از این لیست پیوندی در واقع یک struct به صورت زیر میباشد :

```
typdef struct Node{
    int data;
    Node * next;
}Node;
```

در سوال باید توابع زیر را پیاده سازی نمایید.

```
Node* CreateNode(int data)
```

این تابع یک node با مقدار date می سازد و اشاره گری به این نود باز میگرداند.

```
void PrintNode(Node* node)
```

این تابع با گرفتن یک Node مقدارش را چاپ می کند.

```
void PrintLinkedList(node* head)
```

این تابع با گرفتن head لینک لیستمون کل لینک لیست را چاپ می کند.

```
void ReversePrint(node* head)
```

این تابع با گرفتن head لینک لیستمون کل لینک لیست را به صورت برعکس چاپ می کند.

```
void push_front(node** head, int data)
```

این تابع ابتدا یک Node با مقدار date درست کرده و سپس آن را به ابتدای لینک لیست اضافه می کند.

```
void push_back(node** head, int date)
```

این تابع با گرفتن date و head لینک لیست، یک Node با مقدار date درست کرده و سپس آن را به انتهای لینک لیست اضافه میکند.

```
void pop_front(node** head)
```

این تابع با گرفتن head لینک لیست، نود ابتدایی اش را (در صورت وجود داشتن) حذف می کند.

```
void pop_back(node** head)
```

این تابع با گرفتن head لینک لیست، نود انتهایی اش را (در صورت وجود داشتن) حذف می کند.

```
void insert(node** head, int index , int data)
```

این تابع با گرفتن head لینک لیست یک نود با مقدار data درست کرده و آن را در موقعیت index گفته شده قرار می دهد. (index مون از صفر شروع شده و حتما حواستان باشد که index خواسته شده در دامنه ی لینک لیستمون باشد در صورت نبودن پیام خطا چاپ شود.) به عنوان مثال

```
1 -> 4 -> 6 -> 7
insert(&head , 2 , 55);
1 -> 4 -> 55 -> 6 -> 7
```

ن

```
void delete(node** head , int index)
```

از این تابع برای حذف کردن یک Node از موقعیت دلخواهی از لیست پیوندی استفاده می شود. (در صورتی که index بیشتر یا مساوی سایز آرایه باشد، باید پیغامی مناسب چاپ شده و از تابع خارج شویم.)

```
int search(node** head ,int data)
```

این تابع جهت پیدا کردن اولین اندیسی از لینک لیست که دارای مقدار data می باشد استفاده می شود. (در صورت پیدا نکردن مقدار -1 را باز میگرداند)

```
void swap(node** head ,int index1, int index2)
```

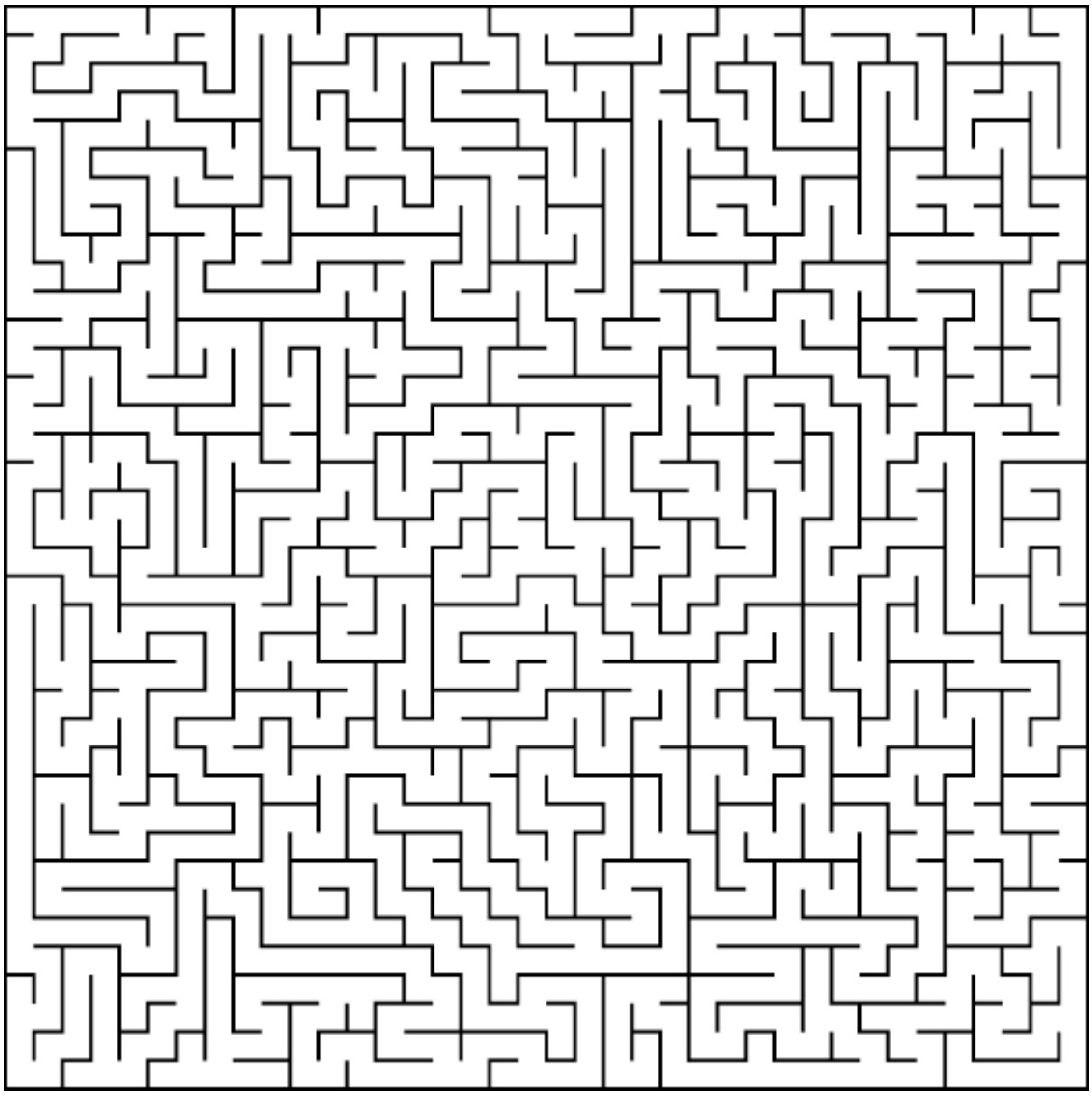
از این تابع برای جا به جایی data مربوط به دو Node با اندیس های index1 و index2 استفاده می شود.

```
void reversed(node** head)
```

این تابع با گرفتن head لینک لیست، لینک لیستمان را برعکس میکند. (دقت شود باید اشاره گر ها به خانه ی بعدی برعکس شوند

(اختیاری) کجا باید برم :(

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: 10 مگابایت

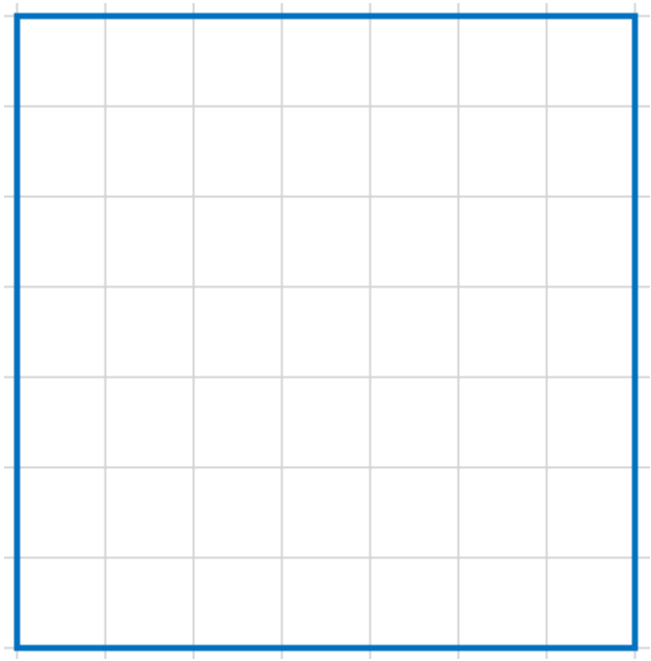


برقی پس از بدبختی و ماکافات فراوان در کمیته انظباطی استفا داد و به یک سفر طولانی اسرار آمیز رفت!! وی در یک از صبح ها وقتی چشم باز می کند خود را در سرزمینی عجیب می یابد بعد از تلاش های فراوان متوجه می

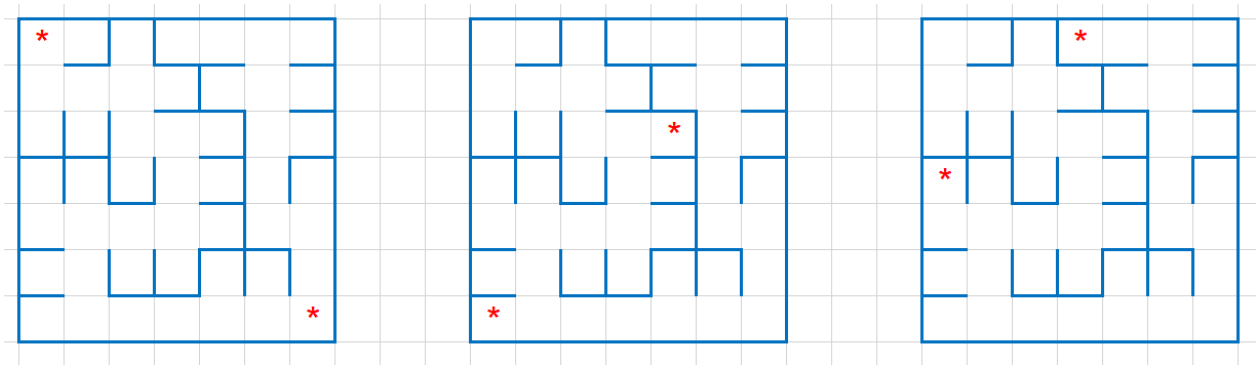
شود که دقیقا در ابتدای یک ماز قرار دارد و تنها راه نجاتش گذر و عبور از این جدول ماز است. پس دست به کار می شود تا راه نجات را پیدا کند. او پس از چندین روز تلاش نا امید می شود اما در می یابد که در مازی با طول m (تعداد ردیف های جدول ماز) و عرض n (تعداد ستون های جدول ماز) گیر افتاده است.در همین حین یادش می افتد که لپ تاپی را با خود به این سفر عجیب و غریب آورده است تا در مسافرت از ددلاین هایش عقب نماند! پس به سرعت سراغ کیفش می رود و با هیجان در میابد که لپ تاپ هنوز سالم هست و کار می کند، اما لپتاپ تنها 15درصد شارژ دارد. پس برقی از شما می خواهد تا هر چه سریعتر به او کمک کنید تا برنامه ای بنویسد که به او برای یافتن نقشه ی احتمالی این ماز کمک کند. برقی می داند در هر روز توانایی t بار بررسی کل ماز را دارد پس باید برنامه ای بنویسد که با گرفتن m (طول ماز) و n (عرض ماز)، t حالت و نقشه متفاوت رندوم از ماز را ترسیم کند تا او بتواند با آن نقشه ها به بررسی ماز بپردازد. نکته ای که باید به آن دقت کنید این است که برقی اصلا فرصت برای بررسی نقشه های تکراری را ندارد و باید هر t نقشه خروجی برنامه شما حداقل در یک خانه تفاوت داشته باشند!

این ماز ویژگی های زیر را باید داشته باشد:

- تمامی دیوارهای داخل ماز، حداقل به یک دیوار متصل هستند و دیواری وجود ندارد که به هیچ دیواری متصل نباشد. همچنین تمامی دیوارها به صورت مستقیم یا غیرمستقیم به دیواره اطراف ماز وصل هستند.
- بین هر دو خانه مربعی ماز، تنها یک مسیر یکتا وجود دارد.



دیواره ی دور ماز در تصویر بالا قابل مشاهده است.



در تصویر بالا مشاهده می شود که بین هر دو خانه در ماز دقیقا یک مسیر یکتا وجود دارد.

این ماز قابل مدل کردن با گراف ها است. یکی از الگوریتم های پیمایش گراف که در حل این مسئله به شما کمک میکند، الگوریتم dfs است که برای فهم آن میتوانید از این لینک و یا سایر منابع موجود در اینترنت استفاده کنید.

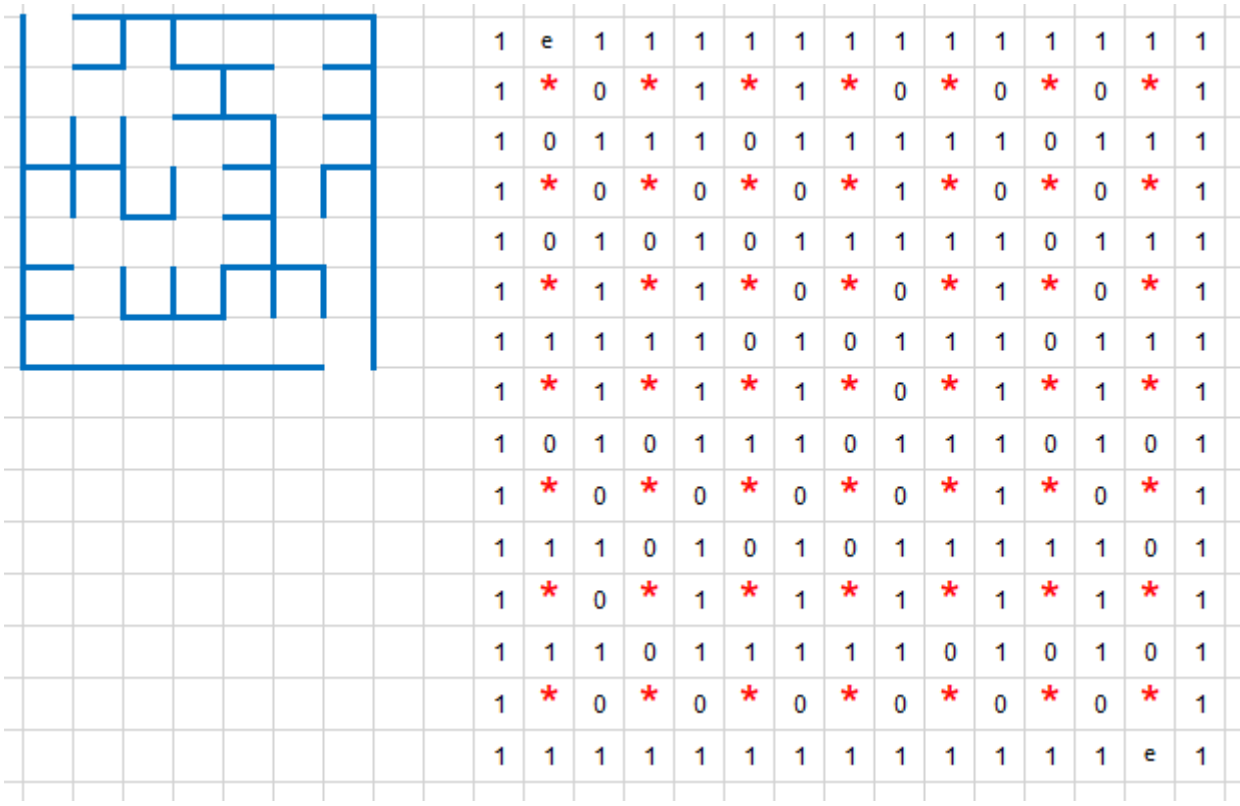
ورودی

ورودی به ترتیب شامل سه عدد m و n و t است. m عرض ماز و n طول آن است. همچنین t تعداد مازهای متفاوتی است که باید در خروجی چاپ شود.

$$1 \leq n, m, t \leq 100$$

خروجی

خروجی به تعداد t جدول به عرض $2 * m + 1$ و طول $2 * n + 1$خواهد بود که هریک به فرمت زیر است. تمامی خانه های ماز (نقاط قرمز رنگ نشان داده شده) می بایست با * در خروجی نشان داده شوند. تمام دیواره های اطراف هر خانه از جدول ماز بدین صورت نشان داده می شوند که در صورت وجود دیوار آن را با `1` و در غیر این صورت (در صورتی که دیواری بین دو خانه وجود نداشت و امکان عبور وجود داشت) آن را با `0` نشان می دهیم.همچنین برای ماز یک ورودی و یک خروجی در نظر گرفته شده است که جای ثابتی دارند که در شکل نیز مشخص است. این ورودی و خروجی ماز با کاراکتر `e` نمایش داده میشوند. در تصویر زیر یک ماز 7*7 و نحوه خروجی دادن آن مشخص شده است:



1	e	1	1	1	1	1	1	1	1	1	1	1	1	1
1	*	0	*	1	*	1	*	0	*	0	*	0	*	1
1	0	1	1	1	0	1	1	1	1	1	0	1	1	1
1	*	0	*	0	*	0	*	1	*	0	*	0	*	1
1	0	1	0	1	0	1	1	1	1	1	0	1	1	1
1	*	1	*	1	*	0	*	0	*	1	*	0	*	1
1	1	1	1	1	0	1	0	1	1	1	0	1	1	1
1	*	1	*	1	*	1	*	0	*	1	*	1	*	1
1	0	1	0	1	1	1	0	1	1	1	0	1	0	1
1	*	0	*	0	*	0	*	0	*	1	*	0	*	1
1	1	1	0	1	0	1	0	1	1	1	1	1	0	1
1	*	0	*	1	*	1	*	1	*	1	*	1	*	1
1	1	1	0	1	1	1	1	1	0	1	0	1	0	1
1	*	0	*	0	*	0	*	0	*	0	*	0	*	1
1	1	1	1	1	1	1	1	1	1	1	1	1	e	1

مثال

ورودی نمونه ۱

```
3 6 3
```

خروجی نمونه ۱

```
1e11111111111
1*0*0*0*0*0*1
1110111111101
1*1*0*1*0*1*1
1011111010101
1*0*0*0*1*0*1
11111111111e1

1e11111111111
1*1*0*0*0*0*1
```

```
1011111110101
1*0*0*0*0*1*1
1111111111101
1*0*0*0*0*0*1
11111111111e1

1e11111111111
1*0*0*1*0*1*1
1011101010101
1*0*1*0*1*0*1
1110111111101
1*0*1*0*0*0*1
11111111111e1
```

توجه داشته باشید که با توجه به اینکه ماز ها به طور رندوم ساخته میشوند، این تنها یک خروجی درست برای این ورودی است.

ورودی نمونه ۲

```
4 11 3
```

خروجی نمونه ۲

```
1e11111111111111111111
1*0*1*0*0*1*1*0*0*1*0*1
11101011101010101011101
1*0*1*1*1*1*0*1*1*0*1*1
10111010101111101110101
1*1*0*1*1*0*1*0*1*0*1*1
10111010111010111011101
1*0*0*1*0*0*0*1*0*0*0*1
11111111111111111111e1

1e11111111111111111111
1*0*0*0*0*0*0*0*0*0*0*1
11111011111011111111101
1*0*1*1*0*0*1*0*0*1*0*1
```

```
10101110111110101110111
1*1*0*1*0*1*0*1*1*0*1*1
11111011101011101011101
1*0*0*0*0*1*0*1*0*0*0*1
11111111111111111111e1

1e11111111111111111111
1*0*1*0*1*0*0*1*0*0*0*1
10101010111010101111101
1*1*0*1*1*0*1*0*1*0*1*1
11111110101111111010101
1*0*1*0*1*1*0*0*0*1*0*1
10101011101011111111111
1*1*0*0*0*1*0*0*0*0*0*1
11111111111111111111e1
```