Problème A: Amis.

Temps : 2sec.

Vous possédez un réseau social. Au total n personnes sont inscrites sur votre réseau ! Comme sur tous les réseaux, chaque personne peut être ami avec d'autres inscrits.

Vous remarquez que certaines personnes sont plus populaires que d'autres. Vous vous demandez donc combien d'amis à la personne la plus populaire de votre réseau social.

Pour vous simplifier la vie, vous attribuer à chaque personne de votre réseau un id de 0 à $n-1$.

Input :

Chaque *test case* sera constitué de :

- ◇ Ligne 1: deux entiers n et r, où n est le nombre de personnes sur votre réseau, avec $2 \leq n \leq 10^3$, et r est le nombre de liens d'amitié, avec $0 < r \leq 10^5$.
- ◇ r lignes constituées de deux entiers a et b distincts, représentant un lien d'amitié entre les personnes a et b (avec $0 \leq a, b < n$). Notez que des liens peuvent apparaître plusieurs fois.

Output :

Pour chaque *test case*, un seul entier, le nombre d'amis distinct de la personne la plus populaire de votre réseau.

Exemple d'input et son output correspondant :

Input 1

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

Output 1

```
2
```

Input 2

```
2 1
0 1
```

Output 2

```
1
```

Problème B: Bots.

Temps : 1sec.

Il est temps pour vous d'évoluer vers une civilisation de type 3 et d'explorer l'espace avec des robots auto-réplicateurs de Von Neumann ! Vos robots de Von Neumann sont capables de deux choses :

- ◇ Prendre une année pour miner une gigatonne de matériau ;
- ◇ Prendre une année pour créer un robot identique à lui-même, capable lui aussi de s'auto-répliquer et/ou de miner des ressources.

Vous avez un objectif de ressources à miner, et un robot auto-réplicateur. En optimisant les instructions données à vos robots, en combien de temps minimum pouvez-vous atteindre votre objectif ?

Input :

Chaque *test case* est constitué d'une unique ligne contenant un seul entier N, l'objectif de matériau à miner, en gigatonne ($1 \leq N \leq 10^9$).

Output :

Pour chaque *test case*, un seul entier, le nombre d'années qu'il vous faudra pour atteindre votre objectif.

Exemple d'input et son output correspondant :

Input 1	**Output 1**
`1`	`1`
Input 2	**Output 2**
`5`	`4`

Explication du test case 2 :

Si votre robot initial ne fait que miner, il lui faudra 5 ans pour miner 5GT. Hors, s'il passe sa première année à se répliquer, vous aurez 2 robots. Ces deux robots peuvent ainsi miner pendant 3 ans, pour récupérer 6GT, et atteindre votre objectif en 4 ans seulement.

Problème C: Caradium.

Temps : 1sec.

Afin d'alimenter votre vaisseau spatial, vous utilisez du Caradium, un alliage de titanium-aluminium-magnesium. L'alliage peut être créé de bien des manières. Si on considère x, y et z comme étant le ratio respectif de titanium, aluminium et magnesium (avec $x + y + z = 1$). La puissance du Caradium en tant qu'alimentation est calculé par $x \times y$, et le coût du Caradium est de $x + y$ méga-euro.

Vous n'avez qu'un montant fixe de méga-euro. Avec ce montant, quel est la plus haute puissance que vous pouvez obtenir pour votre Caradium ?

Input :

Chaque *test case* est constitué d'une unique ligne contenant un nombre décimal c, le nombre de méga-euro que vous possédez, avec $0 \leq c \leq 2$.

Output :

Pour chaque *test case*, un nombre décimal, la plus haute puissance que vous pouvez obtenir.

Exemple d'input et son output correspondant :

Input 1	**Output 1**
`1.0`	`0.25`
Input 2	**Output 2**
`0.2`	`0.01`

Problème D: Dé-Connecté.

Dans des villes toujours en travaux, certaines routes sont impraticables. Il est donc impossible de se rendre de certains endroits à d'autres. Mais on souhaite que la ville puisse fonctionner de nouveau normalement le plus rapidement possible. Il est donc nécessaire de déterminer le nombre minimum de travaux à terminer, et donc de routes à réouvrir, afin qu'il soit possible de se déplacer entre toute paire de points. Dans les villes concernées, la construction d'une route coûte le même prix que la construction d'une autre route, et toutes les routes sont bidirectionnelles.

Pour différentes villes, il vous est demandé de trouver le nombre minimum de routes à ajouter au réseau actuel pour que tout le monde puisse se rendre partout.

Input :

Chaque *test case* sera constitué de :

- ◇ Ligne 1: deux entiers n et r, où n est le nombre de personnes sur votre réseau, avec $2 \leq n \leq 10^3$, et r est le nombre de liens d'amitié, avec $0 < r \leq 10^5$.
- ◇ r lignes constituées de deux entiers a et b distincts, représentant un lien d'amitié entre les personnes a et b (avec $0 \leq a, b < n$). Notez que des liens peuvent apparaître plusieurs fois.

Output :

Pour chaque test case, dans l'ordre du fichier input, un seul entier par ligne représentant le nombre minimum de routes à ajouter.

Exemple d'input et son output correspondant :

Input 1

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

Output 1

```
1
```

Input 2

```
2 1
0 1
```

Output 2

```
0
```

Problème E: Ecole. *(Bonus)*

Temps : 2sec.

Demain, c'est votre premier cours intergalactique ! L'école se trouve loin dans la galaxie, mais la fédération de l'espace a créé des portails de téléportation entre différentes planètes, simplifiant grandement le voyage d'un point à l'autre de l'espace.

Avant d'aller en cours, il vous faut cependant aller chercher votre matériel quantique. Dû à sa nature quantique, cet équipement à une probabilité de se trouver sur chacune des planètes de la galaxie.

Vous devez prévoir l'heure à laquelle vous devez vous lever demain, mais vous souhaitez aussi dormir le plus longtemps possible. Sachant que le temps pour prendre un portail de téléportation est d'une minute, vous souhaitez calculer alors le temps moyen qu'il vous faudra pour aller à l'école demain.

On considèrera toujours que la planète de départ est la planète 1 et que votre école se trouve sur la planète n, la dernière planète de la galaxie.

Input :

Chaque *test case* sera constitué de :

- Ligne 1: deux entiers n et m, le nombre de planètes de la galaxie et le nombre de portails entre les planètes, avec $2 \leq n, m \leq 10^4$.
- Ligne 2: n nombres décimaux p_i, séparés par un espace, où p_i est la probabilité que votre matériel quantique se trouve sur la planète i. Il est garanti que $\sum_i p_i = 1$.
- m lignes contenant deux entiers distincts i et j $(1 \leq i, j \leq n)$, indiquant qu'il existe un portail de téléportation entre les planètes i et j.

Output :

Une ligne contenant le temps moyen qu'il vous faudra pour arriver à l'école demain, à 10^{-6} prêt.

Exemple d'input et son output correspondant :

Input 1

```
3 2
0.0 1.0 0.0
1 2
2 3
```

Output 1

```
2
```

Input 2

```
3 3
0.4 0.6 0.0
1 3
1 2
3 2
```

Output 2

```
1.600
```

Problème F: Factorielle Inverse. *(Bonus)*

Temps : 4sec.

Il est facile de calculer la factorielle d'un petit nombre entier, et vous l'avez probablement déjà fait de nombreuses fois. Dans ce problème, votre tâche est inversée. On vous donne la valeur de $n!$ et vous devez trouver la valeur de n.

Input :

Chaque *test case* sera constitué d'une ligne contenant $n!$, la factorielle d'un entier positif n supérieur à 1. Le nombre de chiffres de $n!$ est au plus 10^6.

Output :

Pour chaque *test case* i, indiquez le nombre n.

Exemple d'input et son output correspondant :

Input 1	**Output 1**
`120`	`5`
Input 2	**Output 2**
`51090942171709440000`	`21`
Input 3	**Output 3**
`1088886945041835216076800000`	`27`

$$300! = \prod_{i=1}^{300} i = 1 \times 2 \times 3 \times ... \times 300$$

$$= 725264321896264299365204576448830388909753943489625436053225980776521270$$
$$306057512216440636035370461297268629388588804173576999416776741259476533$$
$$176716867465515291422477573349939147888701726368864263907759003154226842$$
$$927906974559841225476930271954604008012215776252176854255965356903506788$$
$$725264321896264299365204576448830388909753943489625436053225980776521270$$
$$822437639449120128678675368305712293681943649956460498166450227716500185$$
$$176546469340112226034729724066333258583506870150169794168850353752137554$$
$$910289126407157154830282284937952636580145235233156936482233436799254594$$
$$095276820608062232812387383880817049600000000000000000000000000000000000$$
$$000000000000000000000000000000000000000000$$

Problème G: Goldbach.

Temps : 4sec.

La conjecture de Goldbach est l'assertion mathématique qui s'énonce comme suit :

Tout nombre entier pair supérieur à 3 peut s'écrire comme la somme de deux nombres premiers.

Formulée en 1742 par Christian Goldbach, c'est l'un des plus vieux problèmes non résolus de la théorie des nombres et des mathématiques. Comme cette conjecture n'a jamais été prouvée et qu'aucun contre-exemple n'a été trouvé, nous allons supposer cette conjecture vraie pour les cas considérés dans cet exercice.

Il peut y avoir plusieurs façons de représenter un nombre pair donné par la somme des nombres premiers. Par exemple $26 = 3 + 23 = 7 + 19 = 13 + 13$.

Input :

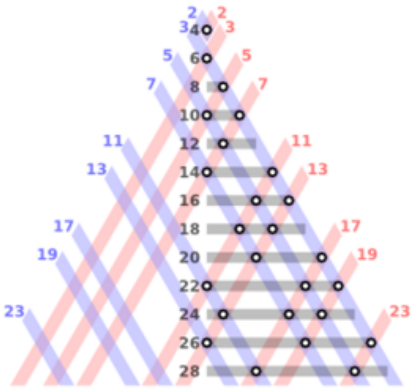
Chaque *test case* sera constitué d'un entier : un nombre x pair compris entre 4 et 10^5.

Output :

Pour chaque *test case*, indiquez **toute** la liste des sommes (une somme par ligne) dans l'ordre croissant de la première addition. Le premier terme de la somme doit toujours être inférieur ou égal au second afin d'éviter les doublons.

Exemple d'input et son output correspondant :

Input 1	**Output 1**
4	2+2
Input 2	**Output 2**
26	3+23 7+19 13+13
Input 3	**Output 3**
100	3+97 11+89 17+83 29+71 41+59 47+53



Problème H: Hackerman.

Temps : 2sec.

Vous êtes un expert en hacking. Votre amie Manon a oublié le mot de passe de son profil Facebook et vous demande de l'aide. Par chance, Manon utilise régulièrement les mêmes mot de passe et, avec son aide, vous arrivez à créer un dictionnaire de mot de passe, et à déduire pour chacun d'eux la probabilité de son utilisation.

Facebook est un site un peu sécurisé, et ne vous autorise à tester qu'un seul mot de passe par seconde. Au vu du dictionnaire de mot de passe, et de leur probabilité respective, combien de temps en moyenne cela vous prendra-t-il pour hacker le mot de passe, en supposant que vous les testiez dans un ordre optimal ? (c'est-à-dire dans l'ordre des probabilités décroissantes).

Input :

Chaque *test case* sera constitué des lignes suivantes :

- ◇ Ligne 1 un entier $n \leq 1000$, le nombre de mots de passe qui composent le dictionnaire.
- ◇ n lignes constituées du mot de passe, un *string* de taille inférieur ou égale à 10 contenant uniquement des caractères alphanumérique (aA-zZ; 0-9), et un *float* x, la probabilité que Manon utilise ce mot de passe, compris entre 0 et 1, contenant au plus quatre décimales. Il est garanti que la somme de tous les x donne exactement 1.

Output :

T lignes : pour chaque *test case* i, le temps moyen en seconde pour hacker le mot de passe en testant les mots de passe dans l'ordre optimal. Votre réponse doit être correcte à 10^{-4} près.

Exemple d'input et son output correspondant :

Input 1

```
3
qwerty 0.5432
123456 0.3334
password 0.1234
```

Output 1

```
1.5802
```

Input 2

```
2
Manon1234 0.5000
1234Manon 0.5000
```

Output 2

```
1.5000
```

Problème I : IJIIJ.

Temps : 2sec.

Vous avez créé au dernier TP la suite `IJIIJ`, fonctionnant de la manière suivante :

$$s_1 = \texttt{J}$$
$$s_2 = \texttt{I}$$
$$s_n = s_{n-1}||s_{n-2}$$

Si nous étendons la suite jusqu'à $N = 5$, les 5 termes sont `J`, `I`, `IJ`, `IJI` et `IJIIJ`. Afin de continuer votre étude de la suite, vous vous demandez cette fois-ci, pour l'élément s_k, combien de caractères `I` et combien de caractères `J` il y a dans cette suite.

Input :

Chaque test case est constitué d'un entier k, avec $1 \leq k \leq 10^6$, le nombre de fois que vous appuyez sur le bouton.

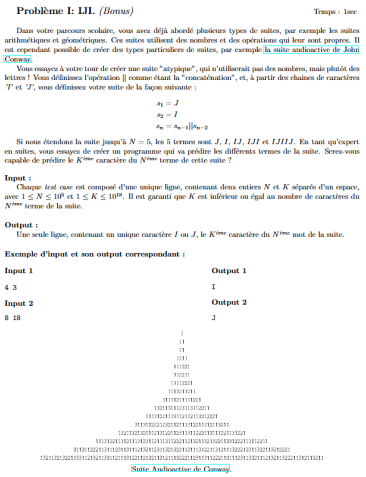
Output :

Pour chaque test case, deux entiers séparés par un espace :

- a, le nombre de caractères `I` modulo $10^9 + 7$ dans s_k, le k^e élément de la suite.
- b, le nombre de caractères `J` modulo $10^9 + 7$ dans s_k, le k^e élément de la suite.

Exemple d'input et son output correspondant :

Input 1	**Output 1**
`1`	`0 1`
Input 2	**Output 2**
`5`	`3 2`
Input 3	**Output 3**
`11`	`55 34`



Problème I: IJI. *(Bonus)* Temps : 1sec

Dans votre parcours scolaire, vous avez déjà abordé plusieurs types de suites, par exemple les suites arithmétiques et géométriques. Ces suites utilisent des nombres et des opérations qui leur sont propres. Il est cependant possible de créer des types particuliers de suites, par exemple la suite audioactive de John Conway.

Vous essayez à votre tour de créer une suite "atypique", qui n'utiliserait pas des nombres, mais plutôt des lettres ! Vous définissez l'opération || comme étant la "concaténation", et, à partir des chaines de caractères 'I' et 'J', vous définissez votre suite de la façon suivante :

$$s_1 = I$$
$$s_2 = I$$
$$s_n = s_{n-1}||s_{n-2}$$

Si nous étendons la suite jusqu'à $N = 5$, les 5 termes sont *J*, *I*, *IJ*, *IJI* et *IJIIJ*. En tant qu'expert en suites, vous essayez de créer un programme qui va prédire les différents termes de la suite. Serez-vous capable de prédire le $K^{ème}$ caractère du $N^{ème}$ terme de cette suite ?

Input :

Chaque *test case* est composé d'une unique ligne, contenant deux entiers N et K séparés d'un espace, avec $1 \leq N \leq 10^6$ et $1 \leq K \leq 10^{18}$. Il est garanti que K est inférieur ou égal au nombre de caractères du $N^{ème}$ terme de la suite.

Output :

Une seule ligne, contenant un unique caractère *I* ou *J*, le $K^{ème}$ caractère du $N^{ème}$ mot de la suite.

Exemple d'input et son output correspondant :

Input 1	**Output 1**
4 3	I
Input 2	**Output 2**
8 18	J

Suite Audioactive de Conway.

Problème J : Jonction.

Temps : 4sec.

On vous fournit un plan de métro un peu particulier : les stations sont indiquées les unes à la suite des autres. Chacun d'entre elle est une jonction, et on vous donne la liste de toutes les autres stations auxquelles la station est reliée. Chaque connexion entre les stations est bidirectionnelle, c'est-à-dire qu'elle peut être empruntée dans les deux sens.

À l'aide de cette carte particulière, et étant données une station de départ et une station d'arrivée, votre tâche consiste à déterminer la séquence de stations à traverser pour atteindre votre destination finale (ou à déclarer qu'il n'y a pas d'itinéraire possible entre les deux stations).

Input :

Chaque *test case* sera constitué de :

- ◇ Ligne 1: un entier N, avec $0 < N \leq 1000$
- ◇ N lignes suivent, décrivent chacune les connexions d'une station de la carte. Chacune de ces lignes commence par le nom de la station qu'elle décrit et est suivie d'une liste séparée par des espaces de tous les noms de stations qui sont directement connectées à cette station.
- ◇ Ligne $N+2$: la dernière ligne identifie une station de départ et une station d'arrivée (différentes l'une de l'autre).

Notez qu'il peut y avoir plus de N stations dans le test case, mais pas plus de $2 * 10^3$ stations.

Output :

Pour chaque *test case* i, indiquez la séquence de stations qui mène de la station de départ à la station d'arrivée. Séparez les noms des stations par des espaces. S'il n'y a pas d'itinéraire possible entre la station de départ et la station d'arrivée, indiquez `pas de route`.

Exemple d'input et son output correspondant :

Input 1

```
6
A B
B A D
C D
E D F G
F E
G E
F A
```

Output 1

```
F E D B A
```

Input 2

```
4
FirstStop SecondStop
SecondStop FirstStop ThirdStop
FifthStop FourthStop SixthStop
SixthStop FifthStop
FirstStop FifthStop
```

Output 2

```
pas de route
```