



Problème Prison

 2 sec.  1024 Mo

Alice et Bob ont été injustement condamnés à purger une peine dans une prison à haute sécurité. Maintenant, ils doivent planifier leur évasion. Pour ce faire, ils doivent être en capacité de communiquer aussi efficacement que possible (en particulier, Alice doit envoyer des informations quotidiennes à Bob). Toutefois, ils ne peuvent pas se rencontrer et peuvent uniquement échanger des informations via des notes écrites sur des serviettes de table. Chaque jour Alice veut envoyer une nouvelle information à Bob : un nombre entre 0 et $N - 1$. À chaque déjeuner, Alice prend trois serviettes et écrit un nombre entre 0 et $M - 1$ sur chaque serviette (il peut y avoir des répétitions) et les laisse sur son siège. Ensuite, leur ennemie, Charly, détruit une des serviettes et mélange les deux autres. Enfin, Bob trouve les deux serviettes restantes et lit les nombres qui y sont inscrits. Il doit correctement décoder le nombre original qu'Alice voulait lui envoyer. Il y a un espace limité sur les serviettes, donc M est fixé. Néanmoins, Alice et Bob ont pour objectif de maximiser l'information transmise, ils peuvent donc choisir N aussi grand qu'ils le souhaitent. Aidez Alice et Bob à implémenter une stratégie pour chacun d'eux de manière à maximiser la valeur de N .



Détails d'implémentation

Puisqu'il s'agit d'un problème de communication, votre programme va être lancé dans deux exécutions séparées (une pour Alice et une pour Bob) qui ne peuvent pas échanger de données ni communiquer d'aucune manière autre que celle décrite ici. Vous devez implémenter trois fonctions:

```
int setup(int M);
```

Cette fonction sera appelée une fois au début de l'exécution d'Alice et une fois au début de l'exécution de Bob. On lui donne M et elle doit renvoyer la valeur de N souhaitée. Les deux appels à `setup` doivent renvoyer le même N .

```
std::vector<int> encode(int A);
```

Cette fonction implémente la stratégie d'Alice. Elle sera appelée avec le nombre à encoder A ($0 \leq A < N$) et doit renvoyer trois nombres W_1, W_2, W_3 ($0 \leq W_i < M$) qui encodent A . Cette fonction va être appelée un total de T fois, une fois par jour (les valeurs de A peuvent se répéter).

```
int decode(int X, int Y);
```

Cette fonction implémente la stratégie de Bob. Elle sera appelée avec deux des trois nombres renvoyés par `encode` dans un ordre quelconque. Elle doit renvoyer la même valeur



A que celle reçue par `encode`. Cette fonction sera aussi appelée T fois, correspondant aux T appels à `encode`, dans le même ordre. Tous les appels à `encode` se produiront avant tous les appels à `decode`.



Contraintes

- $M \leq 4300$
- $T = 5000$



Notation

Pour une sous-tâche spécifique, la fraction S des points que vous recevrez dépend du plus petit N renvoyé par `setup` sur les tests de la sous-tâche. Elle dépend aussi de N^* , qui est la valeur cible de N nécessaire pour obtenir tous les points sur la sous-tâche:

- Si votre solution échoue sur n'importe quel test, alors $S = 0$.
- Si $N \geq N^*$, alors $S = 1.0$.
- Si $N < N^*$, alors $S = \max \left(0.35 \max \left(\frac{\log(N) - 0.985 \log(M)}{\log(N^*) - 0.985 \log(M)}, 0.0 \right)^{0.3} + 0.65 \left(\frac{N}{N^*} \right)^{2.4}, 0.01 \right)$.



Subtasks

Sous-tâche	Points	M	N^*
1	10	700	82017
2	10	1100	202217
3	10	1500	375751
4	10	1900	602617
5	10	2300	882817
6	10	2700	1216351
7	10	3100	1603217
8	10	3500	2043417
9	10	3900	2536951
10	10	4300	3083817



Exemple

Considérez l'exemple suivant avec $T = 5$. Ici, on a un encodage où Alice envoie trois nombres égaux pour encoder 0 ou trois nombres distincts pour encoder 1. Remarquez

que Bob peut décoder le nombre initial avec n'importe quelle paire de nombres envoyés par Alice.

Exécution	Appel de fonction	Valeur de retour
Alice	<code>setup(10)</code>	2
Bob	<code>setup(10)</code>	2
Alice	<code>encode(0)</code>	{5, 5, 5}
Alice	<code>encode(1)</code>	{8, 3, 7}
Alice	<code>encode(1)</code>	{0, 3, 1}
Alice	<code>encode(0)</code>	{7, 7, 7}
Alice	<code>encode(1)</code>	{6, 2, 0}
Bob	<code>decode(5, 5)</code>	0
Bob	<code>decode(8, 7)</code>	1
Bob	<code>decode(3, 0)</code>	1
Bob	<code>decode(7, 7)</code>	0
Bob	<code>decode(2, 0)</code>	1

Évaluateur d'exemple

Pour l'évaluateur d'exemple, tous les appels à `encode` et `decode` vont être dans la même exécution de votre programme. De plus, `setup` va être appelé une seule fois (contrairement à deux fois, une par exécution, dans l'évaluateur officiel).

L'entrée est juste constituée d'un unique entier, M . Ensuite sera affiché la valeur de N que votre `setup` a renvoyé. Ensuite, les fonctions `encode` et `decode` seront appelées T fois, dans cet ordre, avec des nombres générés aléatoirement entre 0 et $N - 1$ et des choix aléatoires pour les deux des trois nombres de `encode` à donner à `decode` (et pour leur ordre). L'évaluateur affichera un message d'erreur si votre solution échoue.