TP n°1 : Introduction à CodeForces

Introduction

Dans ce TP, nous allons utiliser la plateforme **Codeforces**, un site en ligne dédié aux concours de programmation et à la pratique d'algorithmes. Chaque problème y est formulé de manière précise avec un format d'entrée et de sortie bien défini. Vous devez :

— lire attentivement l'énoncé du problème ;

— écrire un programme qui lit l'entrée standard (`stdin`) et produit la sortie attendue sur la sortie standard (`stdout`) ;

— soumettre votre code sur le site pour qu'il soit testé automatiquement sur plusieurs jeux de tests.

Le langage de programmation est **libre** : Codeforces accepte une grande variété de langages (C++, Python, Java, etc.). Cependant, il est conseillé d'utiliser **Python** (facile à prendre en main, syntaxe concise) ou **C++** (rapide et puissant, très utilisé en compétition). Ce TP est une bonne occasion de découvrir le C++ si vous ne l'avez jamais pratiqué.

Lors de la soumission, vous devrez aussi choisir un **compilateur/interpréteur** adapté :

— pour C++, il est recommandé de sélectionner `GNU G++17` ou `GNU G++20` ;

— pour Python, choisissez `Python3` ou `PyPy`.

À titre d'ordre de grandeur, sur Codeforces, un programme en C++ peut exécuter environ 10^8 opérations en 1 seconde, tandis qu'en Python ce chiffre est plutôt autour de 10^7. Cela doit être pris en compte pour choisir vos algorithmes.

Le juge en ligne vérifiera ensuite automatiquement que votre solution est correcte et respecte les contraintes de temps et de mémoire.

Un premier exemple

Exercice 0: Parité

On vous donne deux entiers n et m. Votre tâche est de décider si $n+m$ est pair ou impair.

Entrée : La première ligne contient un entier t ($1 \leq t \leq 10^4$), le nombre de cas de test. Chaque ligne suivante contient deux entiers n et m ($0 \leq n, m \leq 10^9$).

Sortie : Pour chaque cas de test, affichez `YES` si n est pair et `NO` sinon.

Code Python

```
t = int(input())
for _ in range(t):
    n, m = map(int,input().split())
    if (n+m) % 2 == 0:
        print("YES")
    else:
        print("NO")
```

Code C++

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t; cin >> t;
    while (t--) {
        long long n;
        long long m;
        cin >> n>>m;
        if ((n+m) % 2 == 0) cout << "YES\n";
        else cout << "NO\n";
    }
    return 0;
}
```

Pour tester vos solutions sur un exemple, vous pouvez créer un fichier de test et exécuter votre programme depuis le terminal. Créez un fichier `input.txt` contenant les données d'entrée.

Exemple de fichier `input.txt`

```
3
2 4
7 4
9 3
```

Sortie attendue

```
YES
NO
YES
```

Python3 Sauvegardez votre code Python dans un fichier, par exemple `solution.py`. Dans le terminal, utilisez la redirection pour envoyer le contenu de `input.txt` à votre programme :

```
python3 solution.py < input.txt
```

Le programme lira les données depuis `input.txt` et affichera la sortie dans le terminal.

C++ Sauvegardez votre code C++ dans un fichier, par exemple `solution.cpp`. Compilez votre programme avec `g++` :

```
g++ solution.cpp -o solution
```

Exécutez le programme en redirigeant l'entrée depuis `input.txt` :

```
./solution < input.txt
```

Le programme affichera la sortie dans le terminal.

Exercice 1: Somme

On vous donne trois entiers a, b et c. Déterminez si l'un d'eux est la somme des deux autres.

Entrée : La première ligne contient un entier t ($1 \leq t \leq 9261$), le nombre de cas de test. La description de chaque cas de test consiste en trois entiers a, b, c ($0 \leq a, b, c \leq 20$).

Sortie : Pour chaque cas de test, affichez `YES` si l'un des nombres est la somme des deux autres, et `NO` sinon.
La réponse peut être donnée dans n'importe quelle casse (par exemple, `yEs`, `yes`, `Yes` et `YES` seront acceptées comme réponses positives).

Exercice 2: Fête des lumières

La ville de Lyon organise chaque année la célèbre **Fête des Lumières**, durant laquelle de nombreux lampadaires et installations lumineuses décorent la ville.
Dans cette modélisation, les lampadaires publics de Lyon sont numérotés de 1 à N et sont tous allumés par défaut. Un groupe de hackers a obtenu la capacité d'inverser l'état de certains lampadaires par groupes. Chaque fois que les hackers utilisent leur programme, un nombre i (qu'ils ne contrôlent pas) est envoyé au système de gestion. Tous les lampadaires numérotés $i, 2i, 3i, \ldots$ (jusqu'à N) changent alors instantanément d'état : ceux qui étaient allumés s'éteignent, et ceux qui étaient éteints s'allument.
Au cours de la nuit, les hackers utilisent leur programme k fois. On souhaite déterminer le **plus grand nombre de lampadaires simultanément éteints à un moment donné**.

Entrée :

- La première ligne contient un entier N, le nombre de lampadaires ($1 \leq N \leq 10^6$).
- La deuxième ligne contient un entier k, le nombre d'utilisations du programme ($1 \leq k \leq 100$).
- Les k lignes suivantes contiennent chacune un entier i ($1 \leq i \leq N$), indiquant quel signal est envoyé au système de gestion.

Sortie : Votre programme doit afficher un entier : le plus grand nombre de lampadaires simultanément éteints au cours de la nuit.

Exercice 3: Nombre magique

Un **nombre magique** est un nombre formé par la concaténation des nombres 1, 14 et 144. Chaque sous-nombre peut être utilisé plusieurs fois. Par exemple, 14144, 141414 et 1411 sont des nombres magiques, mais 1444, 514 et 414 ne le sont pas. On vous donne un entier n. Déterminez si n est un nombre magique.

Entrée : La première ligne contient un entier n ($1 \leq n \leq 10^9$). Ce nombre ne contient pas de zéro au début.

Sortie : Affichez `YES` si n est un nombre magique, sinon `NO`.

Exercice 4: Le roi s'échappe !

Alice et Bob jouent aux échecs sur un très grand échiquier de dimension $n \times n$. Alice ne possède plus qu'une seule pièce — une reine située en (a_x, a_y), tandis que Bob a uniquement son roi en (b_x, b_y).
Alice pense que sa reine domine l'échiquier et que la victoire lui est assurée.
Mais Bob a un plan rusé pour remporter la partie : il doit amener son roi en (c_x, c_y) pour revendiquer la victoire. Comme Alice est distraite par sa supériorité, elle ne déplace plus aucune pièce, et seul Bob effectue des mouvements.
Bob gagne si son roi peut se déplacer de (b_x, b_y) à (c_x, c_y) sans jamais être en échec. Rappel : le roi peut se déplacer sur n'importe quelle case adjacente parmi les 8 autour de lui. Un roi est en échec s'il se trouve sur la même rangée, colonne ou diagonale qu'une reine ennemie.
Entrée :

- La première ligne contient un entier n ($3 \leq n \leq 1000$), la dimension de l'échiquier.
- La deuxième ligne contient deux entiers a_x et a_y ($1 \leq a_x, a_y \leq n$), la position de la reine.
- La troisième ligne contient deux entiers b_x et b_y ($1 \leq b_x, b_y \leq n$), la position du roi.
- La quatrième ligne contient deux entiers c_x et c_y ($1 \leq c_x, c_y \leq n$), la position que le roi souhaite atteindre.

Il est garanti que le roi n'est pas initialement en échec et que la position cible n'est pas en échec non plus. De plus, le roi et la reine ne se trouvent pas sur la même case, et la cible ne coïncide ni avec la reine ni avec la position initiale du roi.
Sortie : Affichez `YES` si Bob peut atteindre (cx, cy) sans jamais être en échec, sinon `NO`. Vous pouvez écrire la réponse dans n'importe quelle casse.

Exercice 5: Football

Petya adore le football. Un jour, alors qu'il regardait un match, il a noté les positions des joueurs sur un papier. Pour simplifier, il représente la situation comme une chaîne composée de zéros et de uns :

- '0' correspond aux joueurs d'une équipe,
- '1' correspond aux joueurs de l'autre équipe.

Si au moins 7 joueurs d'une même équipe se trouvent côte à côte, la situation est considérée comme **dangereuse**.
Par exemple, la situation '00100110111111101' est dangereuse, tandis que '11110111011101' ne l'est pas.
On vous donne la situation actuelle. Déterminez si elle est dangereuse ou non.

Entrée : Une chaîne non vide composée uniquement de '0' et '1'. La longueur de la chaîne ne dépasse pas 100 caractères. Il y a au moins un joueur de chaque équipe.

Sortie : Affichez `YES` si la situation est dangereuse, sinon `NO`.

Exercice 6: Alice s'envole

Alice va voler vers Mars. Elle doit construire une fusée composée de plusieurs étages disposés dans un certain ordre. Chaque étage est représenté par une lettre minuscule. Ainsi, la fusée peut être décrite par une chaîne de lettres correspondant aux étages.

Il y a n étages disponibles. La fusée doit en contenir exactement k. Les étages dans la fusée doivent être ordonnés selon leur poids : après un étage représenté par une lettre, on ne peut utiliser qu'un étage dont la lettre est au moins deux positions plus loin dans l'alphabet (on peut sauter une ou plusieurs lettres).
Par exemple, après la lettre 'c', on ne peut pas placer 'a', 'b', 'c' ou 'd', mais on peut placer 'e', 'f', ..., 'z'.
Pour que la fusée aille le plus loin possible, son poids total doit être minimal. Le poids d'un étage est son rang dans l'alphabet : 'a' = 1, 'b' = 2, ..., 'z' = 26.
Construisez la fusée de poids minimal ou déterminez qu'il est impossible de construire une fusée.

Entrée :

- La première ligne contient deux entiers n et k ($1 \leq k \leq n \leq 50$), le nombre d'étages disponibles et le nombre d'étages à utiliser.
- La deuxième ligne contient une chaîne s de longueur n, composée de lettres minuscules, représentant les étages. Chaque étage peut être utilisé au plus une fois.

Sortie : Affichez un entier : le poids minimal de la fusée, ou '-1' si la construction est impossible.

Exercice 7: Poursuite (Bonus)

Vous et votre ami Ilya participez à un concours de programmation individuel comprenant plusieurs étapes. Chaque étape rapporte entre 0 et 100 points.
Pour calculer le résultat total après k étapes terminées, on sélectionne les $k - \lfloor k/4 \rfloor$ meilleures étapes et on additionne leurs scores.
Par exemple, si 9 étapes sont terminées et vos scores sont : `50 30 50 50 100 10 30 100 50`, on choisit les 7 meilleures étapes et le total devient 430.
Vous connaissez vos scores et ceux d'Ilya après n étapes, mais le nombre de prochaines étapes est inconnu. Vous devez déterminer le **plus petit nombre d'étapes supplémentaires** à jouer afin que votre résultat puisse devenir supérieur ou égal à celui d'Ilya.

Entrée :

— La première ligne contient un entier t ($1 \leq t \leq 1000$) — le nombre de cas de test.
— Chaque cas de test contient trois lignes :
  — La première ligne : un entier n ($1 \leq n \leq 10^5$) — le nombre d'étapes terminées.
  — La deuxième ligne : n entiers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq 100$) — vos points.
  — La troisième ligne : n entiers $b_1, b_2, \ldots, b_n$ ($0 \leq b_i \leq 100$) — les points d'Ilya.

Il est garanti que la somme des n sur tous les cas de test ne dépasse pas 10^5.

Sortie : Pour chaque cas de test, affichez un entier : le nombre minimal d'étapes supplémentaires nécessaires pour que votre résultat puisse devenir supérieur ou égal à celui d'Ilya. Si votre résultat est déjà supérieur ou égal, affichez 0.