TP spécial : Concours de programmation

Le but de ce TP est de participer à un concours blanc d'algorithmique, type Google Code Jam ou SWERC. Par équipe de 3 ou 4, il faut se connecter sur le serveur et soumettre vos codes qui résolvent les exercices donnés ci-après.
Les consignes pour pouvoir soumettre vos solutions sont données ci-dessous.

Créer un compte sur le serveur vdjudge : (si vous n'en avez pas déjà un...)
- Enregistrez vous sur https://vjudge.net/ (un compte par groupe est suffisant).
- Cliquez sur *'Register'* en haut à droite de la page.
- Créez votre compte.
- Validez votre compte grâce au mail que vous avez reçu.

Participer au concours :
- Cliquez sur *'Contest'* en haut de la page, choisissez *'All Contests'* dans le menu de gauche puis cherchez *'ExoAlgo503'* dans la barre de recherche.
- Pour soumettre votre code pour un exercice, cliquez sur *'Submit'*, choisissez *'Python 3 (pypy 7.3)'* comme langage (si votre programme est effectivement en Python) et copier coller votre code.
- Voyez vos résultats, fermez la fenêtre de dialogue et continuez

Gérer les entrées/sorties pour les différents problèmes :
Le serveur exécute votre code avec des fichiers de données passés sur l'entrée standard. La réponse doit être fournie sur la sortie standard dans le format attendu. Voici quelques indications pour gérer cela.
- Pour faire tourner votre code sur un des exemples donné, il faut mettre les données dans un fichier texte, par exemple `instance.txt`. Ensuite dans un terminal, on appelle `» python3 monCode.py < instance.txt`
- Pour récupérer l'entrée standard dans votre code, on peut s'y prendre de la façon suivante :
  - ▷ Inclure `from sys import stdin`
  - ▷ Parcourir toutes lignes de l'entrée standard : `for ligne in stdin:`
  - ▷ Pour découper une ligne aux espaces et recupérer les différents mots dans une liste : `tab=ligne.split()`
  - ▷ Pour parcourir cette liste : `for w in tab:` (ou `tab[1:]` si on ne veut pas le premier caractère par exemple)
  - ▷ `w` est une chaîne de caractères, il faut le caster si besoin (par exemple `int(w)`)
- Enfin pour écrire sur la sortie standard, on fait simplement
  - ▷ `print()`, en respectant le format attendu (espace, saut de ligne...)

Exercice 1. **Sandwich**

Pépito commence toujours sa matinée avec un bon vieux sandwich. Les sandwiches que Pépito prépare consistent en du pain, du fromage et/ou du jambon.
Un sandwich suit toujours la formule suivante :

- une tranche de pain
- une tranche de fromage ou de jambon
- une tranche de pain

...

- une tranche de fromage ou de jambon
- une tranche de pain

Il a toujours du pain en haut et en bas, et le sandwich alterne entre du pain et de la garniture, où la garniture est une tranche de fromage ou de jambon. Chaque tranche, que ce soit de pain, de fromage ou de jambon, est appelée une couche. Aujourd'hui, Pépito s'est réveillé et a découvert qu'il a b tranches de pain, c tranches de fromage et h tranches de jambon. Quel est le nombre maximum de couches que peut avoir son sandwich du matin ?

Entrée :
La première ligne contient un entier t ($1 \leq t \leq 1000$) ; le nombre de tests.
Chaque test se compose de trois entiers b, c et h avec $2 \leq b \leq 100$ et $1 \leq c, h \leq 100$; le nombre de tranches de pain, de tranches de fromage et de tranches de jambon, respectivement.

Sortie :
Pour chaque test, imprimez un seul entier : le nombre maximum de couches que peut avoir le sandwich du matin de Pépito.

Exemple d'entrée :

```
3
2 1 1
10 1 2
3 7 8
```

Sortie correspondante :

```
3
7
5
```

Notes :
Dans le premier cas de test, Pépito peut se faire un sandwich avec trois couches : une tranche de pain, une tranche de fromage et une autre tranche de pain, ou alors une tranche de pain, une tranche de jambon et une autre tranche de pain.
Dans le deuxième cas de test, Pépito a beaucoup de pain, mais pas assez de garniture. Il peut se faire un sandwich avec quatre tranches de pain, une tranche de fromage et deux tranches de jambon.
Dans le troisième cas de test, c'est l'inverse, Pépito a beaucoup de garniture, mais pas assez de pain. Il peut se préparer un sandwich avec trois tranches de pain et deux tranches de fromage, par exemple.

Exercice 2. **Partition préférée**

Pépito étant rassasié, il s'attaque à son premier exercice d'algorithmique de la journée.
Il souhaite représenter son entier n préféré comme une somme de trois entiers distincts positifs x, y et z. De plus, Pépito ne veut pas que l'un des nombres x, y et z soit divisible par 3.
Votre tâche est d'aider Pépito à trouver n'importe quel triplet valide d'entiers distincts positifs x, y et z, ou de signaler qu'un tel triplet n'existe pas.

Entrée :
La première ligne contient un seul entier t ($1 \leq t \leq 10^4$) : le nombre de cas de test.
La seule ligne de chaque cas de test contient un seul entier n ($1 \leq n \leq 10^9$).

Sortie :
Pour chaque cas de test, s'il n'y a pas de triplet valide x, y et z, imprimez NO sur la première ligne.
Sinon, imprimez YES sur la première ligne. Sur la deuxième ligne, imprimez n'importe quel triplet valide d'entiers distincts positifs x, y et z tels que $x + y + z = n$, et aucun des nombres imprimés n'est divisible par 3. S'il existe plusieurs triplets valides, vous pouvez en imprimer n'importe lequel.

Exemple d'entrée :

```
4
10
4
15
9
```

Sortie correspondante :

```
YES
4 5 1
NO
YES
2 8 5
NO
```

Notes :
Dans le premier cas de test, l'un des triplets valides est $x = 4$, $y = 5$, $z = 1$. Ces trois nombres sont distincts, aucun de ces nombres n'est divisible par trois, et $4 + 5 + 1 = 10$.
Dans le deuxième cas de test, il n'y a pas de triplet valide.
Dans le troisième cas de test, l'un des triplets valides est $x = 2$, $y = 8$, $z = 5$. Ces trois nombres sont distincts, aucun de ces nombres n'est divisible par trois, et $2 + 8 + 5 = 15$.
Dans le quatrième cas de test, il n'y a pas de triplet valide.

Exercice 3. **Tout travail mérite salaire**

Récoltant là le fruit de leur dur labeur, Pépito et ses accolytes se retrouvent à la tête d'un butin constitué de n objets dont les valeurs sont données dans le tableau $[a_1, a_2, \ldots, a_n]$, où toutes les valeurs sont distinctes. Pour chacun de ses k accolytes, Pépito peut lui donner :

- soit les les deux objets de valeur minimum dans le tableau (puis donc les supprimer du tableau) ;
- soit l'objet de valeur maximum dans le tableau (puis le supprimer du tableau).

Pépito étant vénal, ils souhaite calculer la somme maximale possible des éléments dans le tableau résultant après ces k opérations.

Entrée
La première ligne contient un entier t avec $1 \leq t \leq 10^4$; le nombre de cas de test.
Chaque cas de test se compose de deux lignes :

— La première ligne contient deux entiers n et k avec $3 \leq n \leq 2 \times 10^5$ et $1 \leq k \leq 99999$; (avec $2k < n$) ; le nombre d'éléments et d'accolytes, respectivement.
— La deuxième ligne contient n entiers $a_1, a_2, \ldots, a_n$ avec $1 \leq a_i \leq 10^9$ (tous les a_i sont distincts) ; les éléments du tableau.

Sortie
Pour chaque cas de test, imprimez un entier : la somme maximale possible des éléments dans le tableau résultant.

Exemple d'entrée :

```
6
5 1
2 5 1 10 6
5 2
2 5 1 10 6
3 1
1 2 3
6 1
15 22 12 10 13 11
6 2
15 22 12 10 13 11
5 1
999999996 999999999 999999997 999999998 999999995
```

Sortie correspondante :

```
21
11
3
62
46
3999999986
```

Notes
Dans le premier cas de test, en appliquant la première opération, le résultat est le suivant :

— Deux minimums sont 1 et 2 ; en les supprimant, le tableau devient [5, 10, 6], avec une somme de 21 ;

— Un maximum est 10 ; en le supprimant, le tableau devient [2, 5, 1, 6], avec une somme de 14.
21 est la meilleure réponse.
Dans le deuxième cas de test, il est optimal d'effacer d'abord deux minimums, puis un maximum.

Exercice 4. **Aquarium**

Pépito est un chat et adore les poissons, c'est pourquoi il a décidé de construire un aquarium. Il a cependant une pièce de corail composée de n colonnes, dont la i-ème mesure a_i unités de hauteur. Il va devoir construire un réservoir autour du corail comme suit :

1. Choisire un entier $h \geq 1$, la hauteur du réservoir. Construire des murs de hauteur h de chaque côté du réservoir.
2. Ensuite, remplire le réservoir avec de l'eau de telle sorte que la hauteur de chaque colonne soit h, à moins que le corail ne soit plus haut que h, auquel cas aucune eau ne doit être ajoutée à cette colonne.

Par exemple, avec $a = [3, 1, 2, 4, 6, 2, 5]$ et une hauteur de $h = 4$, il utilisera un total de $w = 8$ unités d'eau, comme illustré.

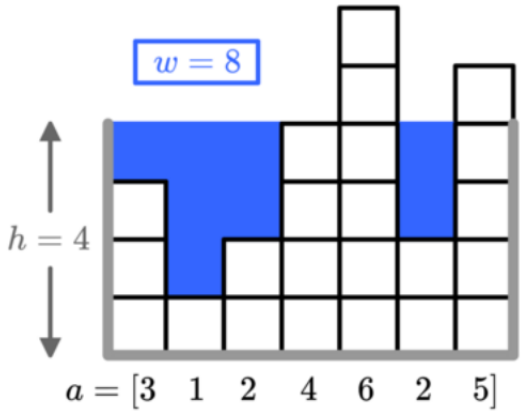

FIGURE 1 – Aquarium avec $a = [3, 1, 2, 4, 6, 2, 5]$ et $h = 4$

Il peut utiliser au plus x unités d'eau pour remplir le réservoir, mais veut construire le plus grand réservoir possible. Quelle est la plus grande valeur de h qu'il peut choisir ?

Entrée :
La première ligne contient un seul entier t avec $1 \leq t \leq 10^4$: le nombre de cas de test.
La première ligne de chaque cas de test contient deux entiers positifs n et x avec $1 \leq n \leq 2 \times 10^5$ et $1 \leq x \leq 10^9$: le nombre de colonnes du corail et la quantité maximale d'eau que tu peux utiliser.
La deuxième ligne de chaque cas de test contient n entiers séparés par des espaces $a_1, \ldots a_n$ avec $1 \leq a_i \leq 10^9$: les hauteurs du corail.
La somme des n sur tous les cas de test ne dépasse pas 2×10^5.

Sortie :
Pour chaque cas de test, affiche un seul entier positif h avec $h \geq 1$: la hauteur maximale que le réservoir peut avoir, de sorte que Pépito n'aie besoin d'au plus x unités d'eau pour remplir le réservoir.
On admettra qu'avec ces contraintes, une telle valeur de h existe toujours.

Exemple d'entrée :

```
5
7 9
3 1 2 4 6 2 5
3 10
1 1 1
4 1
1 4 3 4
6 1984
2 6 5 9 1 8
1 1000000000
1
```

Sortie correspondante :

```
4
4
2
335
1000000001
```

Note :
Le premier cas de test est illustré dans l'énoncé. Avec $h = 4$, nous avons besoin de 8 unités d'eau, mais si h est augmenté à 5, nous avons besoin de 13 unités d'eau, ce qui est plus que $x = 9$. Donc $h = 4$ est optimal.
Dans le deuxième cas de test, nous pouvons choisir $h = 4$ et ajouter 3 unités à chaque colonne, utilisant un total de 9 unités d'eau. On peut montrer que c'est optimal.
Dans le troisième cas de test, nous pouvons choisir $h = 2$ et utiliser toute notre eau, donc c'est optimal.

Exercice 5. **La forêt enchantée**

Marisa vient ramasser des champignons dans la Forêt Enchantée.
La forêt enchantée peut être représentée par n points sur l'axe X, numérotés de 1 à n. Avant que Marisa ne commence, son amie Patchouli a utilisé la magie pour détecter le nombre initial de champignons sur chaque point, représenté par $a_1, a_2, \ldots, a_n$.
Marisa peut commencer à n'importe quel point de la forêt à la minute 0.
À chaque minute, les événements suivants se produisent dans l'ordre :
Elle se déplace du point x à y ($|x - y| \leq 1$, éventuellement $y = x$). Elle collecte tous les champignons sur le point y. Un nouveau champignon apparaît sur chaque point de la forêt.
Notez qu'elle ne peut pas collecter de champignons à la minute 0.
Maintenant, Marisa veut connaître le nombre maximum de champignons qu'elle peut ramasser après k minutes.

Entrée
Chaque test contient plusieurs cas de test. La première ligne contient un entier t avec $1 \leq t \leq 10^4$: le nombre de cas de test. La description des cas de test suit.
La première ligne de chaque cas de test contient deux entiers n, k avec $1 \leq n \leq 2 \times 10^5$ et $1 \leq k \leq 10^9$: le nombre de positions avec des champignons et le temps dont Marisa dispose, respectivement.
La deuxième ligne de chaque cas de test contient n entiers $a_1, a_2, \ldots, a_n$ avec $1 \leq a_i \leq 10^9$: le nombre initial de champignons sur les points 1, 2, …, n.
Il est garanti que la somme des n pour tous les cas de test ne dépasse pas 2×10^5.

Sortie
Pour chaque cas de test, imprimez le nombre maximum de champignons que Marisa peut ramasser après k minutes.

Exemple d'entrée :

```
4
5 2
5 6 1 2 3
5 7
5 6 1 2 3
1 2
999999
5 70000
1000000000 1000000000 1000000000 1000000000
1000000000
```

Sortie correspondante :

```
12
37
1000000
5000349985
```

Notes :

Cas de test 1 :

Marisa peut commencer à $x = 2$. À la première minute, elle se déplace à $x = 1$ et collecte 5 champignons. Le nombre de champignons sera [1, 7, 2, 3, 4]. À la deuxième minute, elle se déplace à $x = 2$ et collecte 7 champignons. Les nombres de champignons seront [2, 1, 3, 4, 5]. Après 2 minutes, Marisa collecte 12 champignons.

Il peut être démontré qu'il est impossible de collecter plus de 12 champignons.

Cas de test 2 :

Voici l'un de ses chemins possibles :

$2 \to 3 \to 2 \to 1 \to 2 \to 3 \to 4 \to 5$

Il peut être démontré qu'il est impossible de collecter plus de 37 champignons.