Concours de programmation

Problèmes du concours

Règlements :

- Pour chaque problème, votre programme doit lire les données d'entrée à partir de l'entrée standard (*clavier*) et les écrire sur la sortie standard (*écran*). Votre programme ne doit pas utiliser de fichiers d'entrée ou de sortie.

- Votre soumission à chaque problème doit consister en un seul fichier source (donc *un code source par problème*).

- Votre code source sera recompilé/interprété avant d'être testé sur des données différentes des vôtres.

- Les sorties de vos programmes doivent correspondre exactement à celles des exemples fournis, incluant l'orthographe et les espacements. Notez que dans les exemples fournis, chaque ligne (y compris la dernière ligne), se termine par un caractère fin-de-ligne (*end-of-line*).

A – Rendre la monnaie[1]

Vous gérez un distributeur automatique de boissons et vous cherchez à minimiser le nombre de pièces de monnaie à rendre lorsqu'un utilisateur met plus d'argent que nécessaire.
Les pièces de monnaie disponibles dans la machine ont des valeurs respectives de : 2$, 1$, 50¢, 25¢, 10¢ et 5¢.
Vous pouvez supposer que le montant `m` fourni par l'utilisateur est toujours supérieur ou égal au prix `p` de l'item acheté dans le distributeur. `m` est donné en ¢ (**cents**). Les commandes n'excèdent jamais 500¢.

En entrée, vous trouverez sur la première ligne un entier `T` indiquant le nombre de jeux de données.
Chaque jeu de données consiste en deux entiers correspondant respectivement au prix `p` de l'item acheté et au montant `m` fourni par l'utilisateur.

En sortie, votre programme doit afficher pour chaque jeu de données le nombre minimal de pièces que le distributeur doit rendre à l'utilisateur.

Exemple d'entrée :

```
2
10 40
50 50
```

Sortie correspondante :

```
2
0
```

1 Source : https://prologin.org/train/2020/semifinal/volvian

B – Ordonner des divinités[2]

Le premier travail du demi-dieu Hercule fut de ramener la peau du lion de Némée. Le terrible animal vivait dans la forêt d'Argolide et terrorisait les habitants de la région. Hercule traversa donc la forêt d'Argolide à la recherche du lion. Là, il vit que des petits panneaux avaient été fixés sur certains arbres. Sur chaque panneau, le nom d'une divinité était inscrit. Hercule nota tous les noms qu'il rencontra. Approchant de l'antre du lion, il vit, gravé dans la pierre :

"La lettre A vaut 1, la lettre B vaut 2, jusqu'à la lettre Z qui vaut 26. Ainsi, le mot : BABA vaut 6 (2+1+2+1). Cherche la valeur de chaque mot, classe-les de la plus petite valeur à la plus grande, et prononce les mots dans cet ordre : le lion se rendra à toi."

L'entrée du problème consiste en un entier `n` sur la première ligne indiquant le nombre de noms de divinités, puis sur la ligne suivante les `n` noms séparés par des espaces.

Avec l'entrée fournie ci-dessous, le mot `ARTEMIS` vaut par exemple `85` et donc il faut le placer avant `ASCLEPIOS` qui vaut `99`.

Entrée du problème :

```
35
ARTEMIS ASCLEPIOS ATHENA ATLAS CHARON CHIRON CRONOS DEMETER EOS ERIS EROS GAIA HADES HECATE HEPHAISTOS HERA HERMES HESTIA HYGIE LETO MAIA METIS MNEMOSYNE NYX OCEANOS OURANOS PAN PERSEPHONE POSEIDON RHADAMANTHE SELENE THEMIS THETIS TRITON ZEUS
```

Sortie correspondante : `sur une seule ligne, les 35 noms de divinités ordonnés selon leurs poids.`

2 Source : https://callicode.fr/pydefis/Herculito01Lion/txt

C – Permutations VALROBSAR[3]

En mathématiques, on étudie souvent les permutations. Une permutation d'une collection d'objets est un arrangement de ces objets dans un certain ordre.
Par exemple, considérons les entiers `1`, `2` et `3`. Il existe 6 façons différentes de les arranger, autrement dit 6 permutations :
`(1,2,3)` `(1,3,2)` `(3,1,2)` `(3,2,1)` `(2,1,3)` `(2,3,1)`

Une permutation est appelée VALROBSAR si aucun entier n'a deux voisins qui sont tous les deux plus petits que lui.

Dans les permutations ci-dessus, `(1,3,2)` et `(2,3,1)` ne sont pas VALROBSAR car dans chacune de ces permutations, l'entier `3` a deux voisins qui sont plus petits que lui.

Par contre les 4 autres permutations : `(1,2,3)`, `(3,1,2)`, `(3,2,1)` et `(2,1,3)` sont VALROBSAR. Par exemple regardons `(3,1,2)` : `3` et `2` n'ont chacun qu'un voisin dont ils ne peuvent pas avoir deux voisins plus petits qu'eux. `1` a deux voisins dont l'un `(3)` est plus grand que lui et l'autre `(2)` est plus petit.

En entrée, on vous donne un seul entier `n`.
En sortie, votre programme doit afficher le nombre de permutations VALROBSAR de la collection `(1,2,3, ..., n)`.

Exemple d'entrée :
3

Sortie correspondante :
4

3 Source : https://cemc.uwaterloo.ca/resources/cemc-at-home/2020/english/grades9-10/20200428_Gr910_MoreCounting-P.pdf

D – L’unique Pokémon[4]

Vous avez réussi à mettre la main sur la liste des positions de tous les Pokémons du globe. Cette liste, relativement longue, précise pour chaque lieu le nom du Pokémon présent, suivi de ses coordonnées (deux nombres entiers séparés par un espace)
Afin d'asseoir votre réputation de chasseur de Pokémons, vous vous êtes mis en tête de capturer le Pokémon unique, qui n'est représenté qu'une seule fois dans la liste.

Vous devez indiquer les coordonnées (deux nombres entiers séparés par un espace) auxquelles vous pourrez le capturer.
Entrez par exemple : `-30 67` si le Pokémon unique a ces coordonnées.

En entrée, vous trouverez sur la première ligne le nombre `n` de Pokémons. Puis sur chacune des `n` lignes suivantes, un triplet de valeurs séparées par des espaces : le nom d’un Pokémon suivi de ses coordonnées.

En sortie, votre programme doit afficher sur une seule ligne, séparées par un espace, les coordonnées du Pokémon unique.

Exemple d'entrée :

```
346
givrali 75 46
branette 35 153
kyogre 23 -10
roserade -97 91
balignon 44 -155
keunotor 32 163
pyronille 11 -102
… Etc. … la liste complète vous est donnée dans le fichier D.in.
```

Sortie correspondante : les coordonnées du Pokémon unique.

4 Source : https://callicode.fr/pydefis/PokePlusRare/txt

E – Nombres riches[5]

Certains nombres entiers ont la propriété suivante :
si on écrit leur carré et leur cube, alors chacun des 10 chiffres est écrit au moins une fois.
Nous appelons ces entiers des *nombres riches*.

C'est le cas de `69`, dont le carré vaut `4761` et le cube `328509`. Chaque chiffre apparaît exactement une fois.

Le nombre `128` est riche aussi. Son carré vaut `16384` et son cube `2097152` (cette fois, le `1` et le `2` sont en double).

En revanche, `42` n'est pas riche. Son carré vaut `1764` et son cube `74088` et les chiffres `2`, `3`, `5`, et `9` manquent à l'appel.

En entrée, on vous donne deux entiers `a` et `b` avec `a < b`.

En sortie, votre programme doit afficher le nombre d'entiers riches trouvés entre `a` et `b` inclus.

Exemple d'entrée :
```
300
500
```

Sortie correspondante :
```
5
```

5 Source : https://callicode.fr/pydefis/NombresRiches/txt

F – Nombre d'îles

Il s'agit ici de compter le nombre d'îles dans une matrice de `0` et de `1` : `0` représente une case d'eau, `1` représente une case de terre. Une île correspond à un ensemble de `1` connectés par 4-voisinage : autrement dit, si un `1` est possède au moins un voisin au nord, au sud, à l'ouest ou à l'est qui est aussi un `1`, alors ils font partie de la même île.

En entrée, vous trouverez sur la première ligne deux entiers `m` et `n` correspondant au nombre de lignes et au nombre de colonnes de la carte (matrice). Puis sur chacune des `m` lignes suivantes, `n` valeurs `0` ou `1` séparées par des espaces.

En sortie, votre programme doit afficher le nombre d'îles trouvées sur la carte.

Exemple d'entrée :

```
8 8
1 1 1 1 0 0 0 0
1 1 1 1 0 0 1 1
1 1 1 1 0 0 1 1
0 0 0 0 0 0 1 1
0 0 1 1 0 0 0 0
0 0 1 1 0 0 0 0
0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1
```

Sortie correspondante :

```
4
```

G– Profit maximal

Étant donné une liste de tâches à compléter avec des dates limites et des profits associés, vous devez maximiser le profit total en sélectionnant les tâches adéquates. On suppose ici que chaque tâche a une durée d'une unité de temps (1) et qu'une seule tâche à la fois est exécutée.

En entrée, vous trouverez sur la première ligne le nombre de tâches `n`. Puis sur chacune des `n` lignes suivantes, trois entiers séparés par des espaces : le numéro de la tâche, sa date limite et son profit.

En sortie, votre programme doit calculer le profit maximal que vous pouvez obtenir.

[FACULTATIF : afficher aussi les numéros des tâches correspondantes.]

Exemple d'entrée :

```
10
1 9 15
2 2 2
3 5 18
4 7 1
5 4 25
6 2 20
7 5 8
8 7 10
9 4 12
10 3 5
```

Sortie correspondante :

```
109
```

Explication : en sélectionnant les tâches `7, 6, 9, 5, 3, 4, 8, et 1 (dans cet ordre)`, on respecte bien les dates limites en obtenant un profit total égal à `109`. Aucune autre combinaison de permet d'obtenir plus de profit.