Lëtzebuerger Informatiksolympiad 2015

Demi-Finale

Problème I CAPITAL 20 points

Tâche

Dans un jeu sur ordinateur le joueur dispose d'un capital initial d'unités d'une devise imaginaire, qu'il peut placer afin de maximiser ses avoirs.

Chaque placement s'élève à exactement une unité et deux effets sont à chaque fois possibles :

- l'unité placée est perdue ;
- l'unité placée est doublée.

Écrire un programme qui détermine à combien d'unités s'élève le capital final du joueur.

Restrictions

Les restrictions suivantes sont à considérer :

- le nombre de jeux J : $1 \leq J \leq 100$,
- le nombre initial d'unités I : $1 \leq I \leq 50$,
- le nombre de placements P effectués : $1 \leq P \leq 100$.

Entrée du programme

Principe

La première donnée est un nombre entier représentant J, le nombre de jeux.

Pour chaque jeu on a :

- un entier désignant le nombre I d'unités du capital initial,
- une chaîne de P caractères, exclusivement des « p » et des « d », sachant que chaque « p » désigne une perte et chaque « d » désigne un doublement. On peut assumer que la chaîne de caractères décrit un jeu possible, c'est-à-dire que le capital ne devient jamais négatif.

Exemple

Le programme traitera deux jeux.

2

10

pddddppdd

10

ddddddddppddd

Sortie du programme

Pour chaque jeu le programme retourne :

- dans une première ligne le texte « Jeu » suivi du numéro du jeu et du symbole « : » ;
- dans une deuxième ligne le nombre d'unités du capital final.

Exemple (correspondant aux données de l'exemple ci-dessus)

Jeu 1:

13

Jeu 2:

20

Remettez le programme sous le nom CAPITAL.*xxx*, avec *xxx*=PAS ou C(PP) ou JAVA.

Le programme doit être développé comme application console exécutable par CAPITAL < *input* > *output* (java CAPITAL < *input* > *output*), *input* et *output* désignant des fichiers de type txt dont le nom peut être quelconque.

Problème II ZOO 40 points

Tâche

La nouvelle attraction du Zoo attire tellement de touristes qu'elle est exposée dans une salle disposant de N places assises numérotées de 1 à N. À l'ouverture et à la fermeture du zoo, la salle est vide.

Au cours de toute la journée, lorsqu'un touriste arrive, on vérifie d'abord si une place est libre.

Si aucune place n'est libre, le touriste doit attendre. Si d'autres touristes arrivent alors qu'il y en a déjà qui attendent, ils doivent se mettre en file correspondant strictement à l'ordre d'arrivée. Le premier de la file occupera la prochaine place libérée.

Si une seule place est libre, le touriste doit occuper cette place.

Si plusieurs places sont libres, le touriste doit occuper la place ayant le plus petit numéro.

La recette par visite sera calculée en fonction de la recette de base de la place attribuée et du poids du touriste, selon lequel on applique un multiplicateur. La durée pendant laquelle le touriste reste à sa place n'est pas considérée.

Écrire un programme qui, connaissant le nombre de touristes M qui visitent la nouvelle attraction ainsi que l'ordre de leurs arrivées et départs, calcule quelle sera la recette totale sur une journée.

Restrictions

Les restrictions suivantes sont à considérer :

- le nombre de places N : $1 \leq N \leq 100$,
- le nombre de touristes M : $1 \leq M \leq 2000$,
- la recette de base applicable à une place donnée R_i : $2 \leq R_i \leq 100$,
- le multiplicateur relatif au poids d'un touriste donné P_j : $2 \leq P_j \leq 10000$.

Observation

Le programme sera testé avec deux types de données en entrée : avec ou sans file d'attente. Tout programme permettant de traiter seulement les jeux de test sans file d'attente sera doté de la moitié des points.

Entrée du programme

Principe

Les données sont entrées comme suit :

- un nombre entier désignant le nombre de places de la salle, avec $N > 0$,
- un nombre entier désignant le nombre de touristes, avec $M > 0$,
- N nombres entiers R_i désignant chacun - pour les places numérotées de 1 à N - le prix applicable à la place numérotée i, avec $1 \leq i \leq N$,
- M nombres entiers P_j désignant chacun le multiplicateur relatif au poids du touriste numéroté j, avec $1 \leq j \leq M$,
- 2 * M nombres entiers indiquant chacun l'arrivée ou le départ d'un touriste j. Un j positif désigne l'arrivée du touriste j, un j négatif (j précédé du symbole -) désigne le départ du touriste j.

Exemple 1 : sans file d'attente

	Explications
3	3 places sont disponibles
4	4 touristes souhaitent entrer
2	La recette de base de la place numérotée 1 est de 2.
3	La recette de base de la place numérotée 2 est de 3.
5	La recette de base de la place numérotée 3 est de 5.
200	Le multiplicateur relatif au poids du touriste numéroté 1 est 200.
100	Le multiplicateur relatif au poids du touriste numéroté 2 est 100.
300	Le multiplicateur relatif au poids du touriste numéroté 3 est 300.
800	Le multiplicateur relatif au poids du touriste numéroté 4 est 800.
3	Le touriste 3 arrive et prend la place 1. (Recette : 2 * 300 = 600)
2	Le touriste 2 arrive et prend la place 2. (Recette : 3 * 100 = 300)
-3	Le touriste 3 quitte et libère la place 1
1	Le touriste 1 arrive et prend la place 1. (Recette : 2 * 200 = 400)
4	Le touriste 4 arrive et prend la place 3. (Recette : 5 * 800 = 4000)
-4	Le touriste 4 quitte et libère la place 3.
-2	Le touriste 2 quitte et libère la place 2.
-1	Le touriste 1 quitte et libère la place 1.

Exemple 2 : avec file d'attente

	Explications
2	2 places sont disponibles
4	4 touristes qui souhaitent entrer
5	La recette de base de la place numérotée 1 est de 5.
2	La recette de base de la place numérotée 2 est de 2.
100	Le multiplicateur relatif au poids du touriste numéroté 1 est 100.
500	Le multiplicateur relatif au poids du touriste numéroté 2 est 500.
1000	Le multiplicateur relatif au poids du touriste numéroté 3 est 1000.
2000	Le multiplicateur relatif au poids du touriste numéroté 4 est 2000.
3	Le touriste 3 arrive et prend la place 1. (Recette : 5 * 1000 = 5000)
1	Le touriste 1 arrive et prend la place 2. (Recette : 2 * 100 = 200)
2	Le touriste 2 arrive et doit attendre en première position.
4	Le touriste 4 arrive et doit attendre en deuxième position.
-1	Le touriste 1 quitte et libère la place 2. Le touriste 2 prend la place 2. (Recette : 2 * 500 = 1000)
-3	Le touriste 3 quitte et libère la place 1. Le touriste 4 prend la place 1. (Recette : 5 * 2000 = 10000)
-2	Le touriste 2 quitte et libère la place 2.
-4	Le touriste 4 quitte et libère la place 1.

Sortie du programme

Le programme retourne la recette totale de la journée.

Exemple 1 (correspondant aux données de l'exemple 1 ci-dessus)

5300

Exemple 2 (correspondant aux données de l'exemple 2 ci-dessus)

16200

Remettez le programme sous le nom ZOO.*xxx*, avec *xxx*=PAS ou C(PP) ou JAVA.

Le programme doit être développé comme application console exécutable par ZOO < *input* > *output* (java ZOO < *input* > *output*), *input* et *output* désignant des fichiers de type txt dont le nom peut être quelconque.

Problème III ATELIERS 40 points

Tâche

Sur un grand campus composé de plusieurs lycées, pour le jour de la fête scolaire, les directions décident d'organiser des ateliers spécifiques à chacun des lycées dans lesquels les élèves peuvent s'inscrire librement. Chaque élève ne peut participer qu'à un seul atelier d'un même lycée.

Votre travail consiste à écrire un programme qui produit pour chaque lycée une liste de tous les ateliers avec le nombre d'élèves inscrits. Les données des lycées sont indépendantes les unes des autres.

Entrée du programme

- Le fichier d'entrée contient les données de plusieurs lycées. Les données d'un lycée sont terminées par une ligne débutant avec le caractère « 1 ».
- La fin du fichier à traiter est signalée par une ligne contenant le caractère « 0 » en première position.
- Les données d'un lycée débutent par une ligne contenant le nom d'un atelier *NA*. Elle est suivie par 0 ou plusieurs lignes contenant chacune un identifiant d'élève *NE* voulant s'inscrire à cet atelier.

Restrictions

Les restrictions suivantes sont à considérer :

- NA est une chaîne de caractères composée uniquement de lettres majuscules « A » à « Z » et de « whitespace » ;
- NE est une chaîne de caractères composée uniquement de lettres minuscules « a » à « z » et de chiffres « 0 » à « 9 », mais débutant par une lettre en minuscule ;
- le nombre d'élèves est au maximum de 1500 ;
- le nombre d'ateliers proposés par un lycée ne dépasse pas 50.

Sortie du programme

- Pour chacun des lycées, le programme affiche la liste des ateliers proposés avec le nombre d'élèves inscrits. L'affichage des données d'un lycée débute par une ligne blanche. Ensuite, chacune des lignes d'une liste débute par le nom de l'atelier suivi d'un caractère « espace » et terminée par le nombre d'élèves inscrits. La liste doit être triée par ordre décroissant du nombre d'élèves inscrits. Si deux ateliers ou davantage ont le même nombre d'inscrits, ils doivent être présentés dans l'ordre alphabétique croissant des noms des ateliers. Si un même élève s'est inscrit plusieurs fois pour le même atelier, il ne doit être compté qu'une seule fois. Par contre, s'il s'est inscrit dans deux ateliers différents ou plus du même lycée, il ne doit pas être pris en compte du tout !

Exemple

Entrée

ATELIER A

schmu123

fatzu234

ATELIER B

ATELIER C

fatzu234

faulen637

happy111

1

ATELIER A

flotten939

lidderech12

schmu123

1

0

Sortie

ATELIER C 2

ATELIER A 1

ATELIER B 0

ATELIER A 3

Remettez le programme sous le nom ATELIERS.*xxx*, avec *xxx*=PAS ou C(PP) ou JAVA.

Le programme doit être développé comme application console exécutable par ATELIERS < *input* > *output* (java ATELIERS < *input* > *output*), *input* et *output* désignant des fichiers de type txt dont le nom peut être quelconque.