Concours Informatique Luxembourgeois 2004
EPREUVE FINALE

Cotation sur 100 points

Problème I - Nombres superpremiers 25 points

Les nombres superpremiers sont des nombres premiers avec la propriété suivante: si on supprime un nombre quelconque de chiffres en partant de la droite, les chiffres restants représentent toujours un nombre premier.

Ainsi, le nombre 5939333 est un nombre superpremier, car les nombres 5939333, 593933, 59393, 5939, 593, 59 et 5 sont tous premiers.

Ecrivez un programme qui recherche tous les nombres superpremiers dans un intervalle donné.

Restrictions

Les bornes inférieures et supérieures de l'intervalle entier positif n'excèdent pas 2147483647 (*pour Turbo Pascal, prendre une variable du type LongInt*).

Entrées et sorties du programme

⇨ Le programme lit de l'entrée standard les bornes de l'intervalle (séparées par un espace).

⇦ Le programme écrit sur la sortie standard, ligne par ligne, les nombres recherchés. S'il n'existe pas de solution, le programme écrit le message "Pas de solution!".

Remettez le programme sous le nom NSP.xxx, avec xxx=PAS ou C(PP). Remettez également le fichier binaire exécutable NSP.EXE correspondant au programme.

Problème II - Expressions arithmétiques entières 25 points

Une expression arithmétique entière est un ensemble ordonné d'éléments tels que nombres entiers, opérateurs, parenthèses et crochets.

Les règles de construction d'une expression arithmétique valide sont les mêmes que celles en mathématiques.

On suppose qu'on dispose des opérateurs binaires suivants:
- \+ pour l'addition
- \- pour la soustraction
- x pour la multiplication
- : pour la division

On dispose en outre de l'opérateur unaire suivant: - pour la négation

Exemples d'expressions arithmétiques entières valides

-7+(-18)
2x8-4x(9)
21+(-3x9):(5-(-7))
13x[2+4x5-(-3x2)-1]x8
[(2+3)]-((-3))

Contre-exemples

17*-4x	deux opérateurs ne peuvent pas se suivre!
(67-[34-13])	les crochets doivent englober les parenthèses, pas inversement!
(1+2(3+4)	l'opérateur * n'est pas implicite!
[3+(19*4]-5)	problème avec l'imbrication des crochets resp. des parenthèses!
-4-(99:7)]	il manque un crochet ouvrant!

Ecrivez un programme qui admet comme entrée le fichier texte EXPRESS.IN contenant une expression arithmétique entière et qui vérifie si une expression arithmétique est valide ou non. Par conséquent, le programme écrit sur sur la sortie standard un des messages "Expression valide" ou "Expression non valide". Le programme ignorera les espaces contenus dans l'expression.

Exemple

EXPRESS.IN -21+(-3x9):[5-(-7)]-1

Sortie standard: Valide

Restrictions

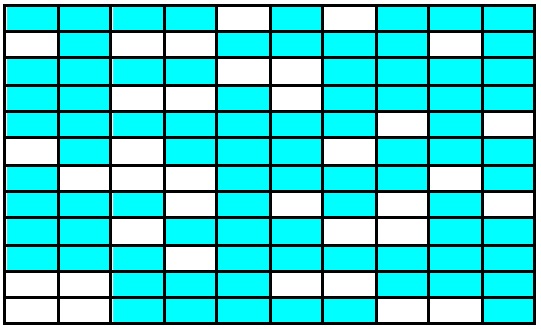
Les expressions arithmétiques entières à vérifier se limitent à 256 caractères et les nombres entiers tombent dans l'intervalle [-32768..32767] (*pour Turbo Pascal, cela représente le domaine du type Integer).*

Remettez le programme sous le nom EXPRESS.xxx, avec xxx=PAS ou C(PP). Remettez également le fichier binaire exécutable EXPRESS.EXE correspondant au programme.

Problème III - Cinéma 50 points

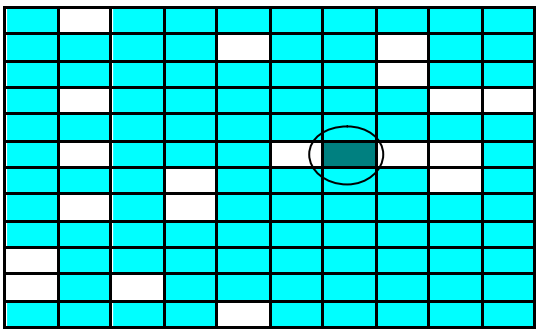
Considérons la situation suivante: 3 copains arrivent dans une salle de cinéma presque pleine. Ils ont du mal à trouver 3 places adjacentes, mais veulent absolument être assis côté à côté. Il faut donc vérifier s'il y a encore 3 places adjacentes libres. Sinon, il faut déterminer si on pouvait obtenir 3 places adjacentes si un ou plusieurs spectateurs pouvaient reculer d'une ou de plusieurs places dans la même rangée. Ecrivez un programme pour aider les trois copains - et toutes les autres personnes qui se situent dans des situations similaires.

Exemple 1



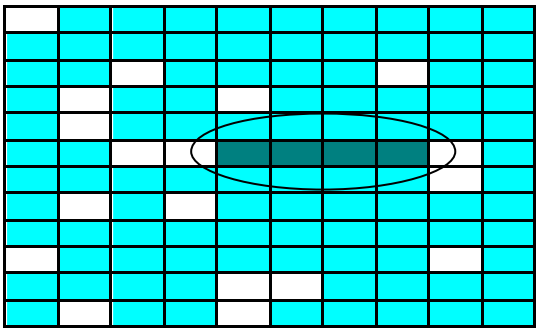
Dans cet exemple, les 3 copains ont de la chance, il y a encore 3 places adjacentes libres.

Exemple 2



Dans cet exemple, les 3 copains ont moins de chance, mais si l'occupant de la place encerclée reculait de 2 places vers la droite ou bien d'une seule place vers la gauche, il y aurait 3 places adjacentes libres.

Exemple 3



Dans cet exemple, si tous les occupants des places encerclées reculaient d'une place vers la droite ou de deux places vers la gauche, il y aurait 3 places adjacentes libres.

Restrictions

- La disposition des places est toujours rectangulaire, de façon qu'elle puisse être représentée avec un tableau à 2 dimensions R×S.

$$8 \leq \text{nombre de rangées } R \leq 40$$
$$8 \leq \text{nombre de places par rangée } S \leq 40$$

- Le nombre de personnes N qui sont à la recherche de places adjacentes libres varie entre 2 et 8.
- Il y a 3 cas de figure qui se présentent:

  1: il y a encore N places adjacentes libres (comme dans l'exemple 1);

  2: il y aurait N places adjacentes libres si un certain nombre de personnes (assises sur des places adjacentes dans une même rangée) reculaient tous ensemble d'une ou plusieurs places vers la gauche ou la droite de la même rangée (comme dans les exemples 2 et 3) - il est inutile d'indiquer la portée et direction du décalage;

  3: il n'y a pas N places adjacentes libres, même si un certain nombre de personnes (assises sur des places adjacentes dans une même rangée) reculaient tous ensemble un ou plusieurs places vers la gauche ou la droite de la même rangée.

Entrées et sorties du programme

⇨ Le programme lit du fichier texte CINE_IN.TXT qui donne le nombre de rangées, le nombre de places par rangée, le nombre de places adjacentes recherché ainsi que l'occupation des places dans le format suivant.

La première ligne contient le nombre de rangées ainsi que le nombre de places par rangée (séparés par un espace).

La deuxième ligne contient le nombre de places adjacentes recherché.

Les lignes suivantes contiennent l'occupation des places d'après la convention suivante: 0 ≡ place libre, 1 ≡ place occupée. Les chiffres sont séparés par un espace. Chaque ligne correspond à une seule rangée.

⇦ Le programme écrit dans le fichier texte CINE_OUT.TXT qui correspond au fichier d'entrée avec les modifications suivantes, correspondant aux trois cas de figure qui se présentent:

1: les N places sont marquées par la lettre 'L' (pour libre);

2: les places des personnes qui doivent reculer sont marquées par la lettre 'D' (pour décalage);

3: le fichier de sortie possède la ligne supplémentaire "Pas de chance!".

Exemple

Le fichier d'entrée correspondant à l'exemple 3 ci-dessus est le suivant:

```
12 10
3
0 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 0 1 1 1 1 0 1 1
1 0 1 1 0 1 1 1 1 1
1 0 1 1 1 1 1 1 1 1
1 1 0 0 1 1 1 1 0 1
1 1 1 1 1 1 1 1 0 1
1 0 1 0 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1 0 1
1 1 1 1 0 0 1 1 1 1
1 0 1 1 0 1 1 1 1 1
```

Le fichier de sortie correspondant à l'exemple 3 ci-dessus est le suivant:

```
12 10
3
0 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 0 1 1 1 1 0 1 1
1 0 1 1 0 1 1 1 1 1
1 0 1 1 1 1 1 1 1 1
1 1 0 0 D D D D 0 1
1 1 1 1 1 1 1 1 0 1
1 0 1 0 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1 0 1
1 1 1 1 0 0 1 1 1 1
1 0 1 1 0 1 1 1 1 1
```

Mise à disposition de données

Dans votre répertoire CILxx, il y a 3 fichiers texte que vous pouvez utiliser en tant que test:

CINE_IN1.TXT - Tableau 25×25; N=4 ➔ Cas de figure 1
CINE_IN2.TXT - Tableau 15×10; N=5 ➔ Cas de figure 2
CINE_IN3.TXT - Tableau 10×20; N=3 ➔ Cas de figure 3

Remettez le programme sous le nom CINE.xxx, avec xxx=PAS ou C(PP). Remettez également le fichier binaire exécutable CINE.EXE correspondant au programme.