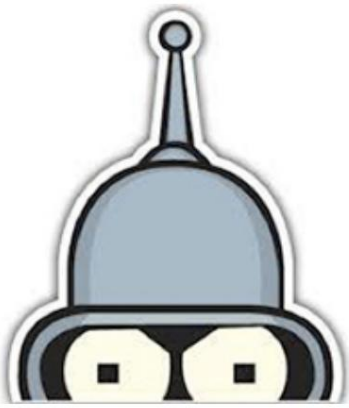
B2 - Programmation élémentaire en C

B-CPE-200

Lem-in

Unité de calcul basée sur Ant



1.1

Lem-in

nom binaire : lem_in nom du
dépôt : CPE_lemin_$ACADEMICYEAR droits du dépôt : langage
ramassage-tek : compilation C : via Makefile, y
compris les règles
re, clean et fclean

- Votre référentiel doit contenir la totalité de vos fichiers source, mais pas de fichiers inutiles (fichiers binaires, temp, obj,...).
- Tous les fichiers bonus (y compris un éventuel Makefile spécifique) doivent être dans un répertoire bonus nommé .
- Les messages d'erreur doivent être écrits sur la sortie d'erreur, et le programme doit alors se terminer avec le code d'erreur 84 (0 s'il n'y a pas d'erreur).

Pour ce projet, les seules fonctions autorisées sont read, write, malloc, free et getline.

> Hex est un ordinateur élaboré, alimenté par la magie et auto-construit (un peu comme la `` pagaille '', une sorte d'appareil magique utilisé par les sorcières du disque-monde) et est logé dans le sous-sol du bâtiment magique à haute énergie de l'Université invisible. (UU) dans la ville jumelle d'Ankh-Morpork.
>
> Hex est un ordinateur pas comme les autres. Programmé via 'Softlore', Hex s'exécute et évolue sous les yeux attentifs du sorcier Ponder Stibbons, qui devient de facto le responsable informatique de l'UU car il est le seul à comprendre de quoi il parle.
>
> Hex a ses origines dans un appareil créé par des étudiants sorciers dans le bâtiment High Energy Magic.
> Sous cette forme, il s'agissait simplement d'un réseau complexe de tubes de verre, contenant des fourmis. Les sorciers pouvaient alors utiliser des cartes perforées pour contrôler les tubes dans lesquels les fourmis pouvaient ramper, leur permettant d'effectuer des fonctions mathématiques simples.
>
> https://en.wikipedia.org/wiki/Hex_(Discworld)

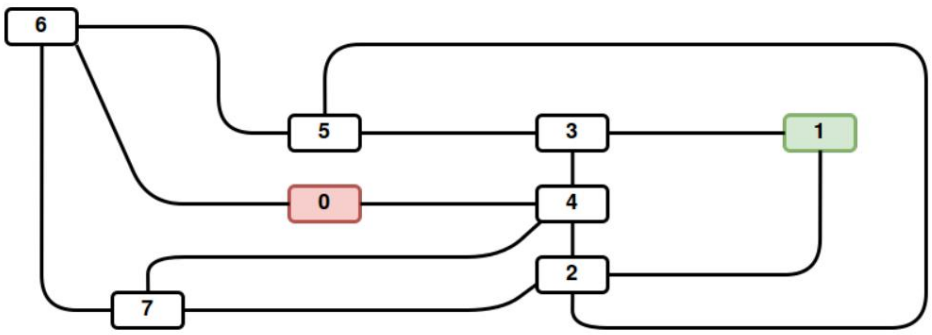
Portons une attention particulière à son unité de calcul : une fourmilière avec chambres et tunnels. Une pièce peut être reliée à un nombre infini d'autres pièces par autant de tunnels que nécessaire, mais un tunnel ne peut en relier que deux pièces.

Ce serait bien d'en construire un à nous, mais comme on n'est pas vraiment bricoleur, créons une version high-tech : un simulateur « Hex'calculation unit ».

Sur l'entrée standard, votre programme doit recevoir la description de la fourmilière de la façon suivante : d'abord, le nombre de fourmis, puis les pièces et leur position et enfin les tunnels (définis par les pièces qu'elles relient).
Voici un exemple d'entrée et la fourmilière associée : Borne

```
/B-CPE-200> entrée chat 34

##commencer
1 23 3
2 16 7
#commentaire
3 16 3
4 16 5
5 9 3
6 1 0
7 4 8
##fin
0 9 5
0-4
0-6
1-3
4-3
5-2
3-5
#un autre commentaire
4-2
2-1
7-6
7-2
7-4
6-5
```



Les noms des pièces ne seront pas nécessairement des numéros dans l'ordre, et sont moins susceptibles d'être ininterrompus (vous pourriez même avoir des pièces nommées : zdfg, 256, qwerty, . . .)

Les coordonnées des salles seront toujours des nombres entiers. Veuillez noter qu'il est possible d'insérer des commentaires en utilisant "#" et des commandes en utilisant "##". ##start indique
que la pièce suivante est l'entrée de la fourmilière et ##end indique que la pièce suivante est la sortie de la fourmilière.

Toutes les commandes inconnues seront ignorées.
L'objectif de votre programme est de trouver le moyen le plus rapide de faire traverser la fourmilière aux fourmis. Pour ce faire, chaque fourmi doit emprunter le chemin le plus court (et pas forcément le plus facile), sans marcher sur ses congénères, et en évitant les embouteillages.

Au début du jeu, toutes les fourmis sont à l'entrée de la fourmilière.
Le but est de les mener jusqu'à la salle de sortie, en un minimum de tours.
Chaque pièce peut contenir une seule fourmi à la fois (sauf ##start et ##end, qui peuvent en contenir autant que nécessaire).

A chaque tour, vous ne pouvez déplacer chaque fourmi qu'une seule fois en suivant un tunnel (si la salle de réception est dégagée).

Vous devez afficher le résultat sur la sortie standard, dans cet ordre : nombre_de_fourmis, pièces, tunnels, puis pour chaque tour, une série de Pn-r où n est le numéro de la fourmi et r le nom de la pièce dans laquelle elle pénètre .
Dans la sortie, vous devez afficher un commentaire indiquant la partie qui va suivre. Ces commentaires doivent être exactement comme dans les exemples.

La première ligne non conforme ou vide entraînera la fin de l'acquisition fourmilière, ainsi que son traitement normal avec les données déjà acquises.

Ce n'est pas aussi simple que de claquer des doigts. Concernant le type d'opération que les étudiants en magie peuvent effectuer avec un tel ordinateur, tout ce que nous savons, aujourd'hui, c'est que l'électricité est plus fiable.

EXEMPLES

Concernant la fourmilière triviale suivante :

[0] - - - [2] - - - [3] - - - [1]

Terminal - + x

```
/B-CPE-200> fourmilière de chat
3
##commencer
0 1 0
##fin
1 13 0 #chambre
2 5 0
# La pièce suivante est la cuisine
3 9 0
0-2
2-3
3-1
```

Terminal - + x

```
/B-CPE-200> ./lem_in < fourmilière
#nombre_de_fourmis
3
#pièces
##commencer
0 1 0
##fin
1 13 0
2 5 0
3 9 0
#tunnels
0-2
2-3
3-1
#se déplace
P1-2
P1-3 P2-2
P1-1 P2-3 P3-2
P2-1 P3-3
P3-1
```

Voilà, c'est fini après 5 tours.

Considérant la fourmilière suivante :

```
      [2] /
    | \ [0] |
[1] \ | / [3]
```

Terminal - + x

```
/B-CPE-200> fourmilière de chat
3
2 5 0
##commencer
0 1 2
##fin
1 9 2
3 5 4
0-2
0-3
2-1
3-1
2-3
```

Terminal - + x

```
/B-CPE-200> ./lem_in < fourmilière
#nombre_de_fourmis
3
#pièces
2 5 0
##commencer
0 1 2
##fin
1 9 2
3 5 4
#tunnels
0-2
0-3
2-1
3-1
2-3
#se déplace
P1-2 P2-3
P1-1 P2-1 P3-2
P3-1
```

C'est fini après 3 tours.

PRIME

En bonus, pourquoi ne pas coder une visionneuse de fourmilière ?

Que ce soit en deux dimensions (une vue de dessus) ou du point de vue d'une fourmi dans un environnement de couloir 3D (avec l'Oculus Rift ?).

Veuillez noter que du fait que les commandes et les commentaires sont répétés sur la sortie du programme, il est possible de passer des commandes spécifiques au visualiseur (pourquoi pas des couleurs, des niveaux, . . . ?).

Un exemple d'utilisation peut être :

Terminal - + x

```
/B-CPE-200> ./lem_in < carte | ./téléspectateur
```