Programmation impérative, projet 2020

Dates et principe

Cette page peut être mise à jour, avec informations complémentaires, précisions, *questions bonus*, etc. Pensez à y revenir souvent.

Projet à rendre pour le **4/1/2020** à **23h59**, aucun retard ne sera toléré.
Des soutenances pourront être organisées ensuite.

Lire tout le sujet.

Un rendu de projet comprend :

- Un rapport typographié précisant vos choix, les problèmes techniques qui se posent et les solutions trouvées ; il précise en conclusion les limites de votre programme. Le rapport sera de préférence composé avec LaTeX. Le soin apporté à la grammaire et à l'orthographe est largement pris en compte.
- Un code *abondamment* commenté ; la première partie des commentaires comportera systématiquement les lignes :
    1. `@ requires` décrivant les préconditions : c'est-à-dire conditions sur les paramètres pour une bonne utilisation (**pas de typage ici**),
    2. `@ assigns` listant les zones de mémoire modifiées,
    3. `@ ensures` décrivant la propriété vraie à la sortie de la fonction lorsque les préconditions sont respectées, le cas échéant avec mention des comportements en cas de succès et en cas d'échec,

    En outre chaque boucle `while` doit contenir un commentaire précisant la raison de sa terminaison (le cas échéant). De même en cas d'appels récursifs.
    On pourra préciser des informations additionnelles si des techniques particulières méritent d'être mentionnées.

    Le code doit enfin compiler sans erreur (évidemment) et sans warning lorsque l'option `-Wall` est utilisée.

    Un code qui ne compile pas se verra attribuer la note de 0.

- Un manuel d'utilisation de votre exécutable, même minimal, est toujours bienvenu.

Avez-vous lu tout le sujet ?

Protocole de dépôt

Vous devez rendre

- Votre rapport (en pdf) et
- Vos fichiers de code

rassemblés dans une archive tar gzippée identifiée comme *votre_prénom_votre_nom*`.tgz`.
La commande devrait ressembler à :
`tar zcvf randolph_carter.tgz rapport.pdf fichiers.c autres_truc_éventuels.c...`
N'OUBLIEZ surtout PAS de mettre le nom identifiant l'archive (donc nouveau) **en PREMIER.**
Lisez le man ! et testez le contenu de votre archive (une commande comme par exemple :
`tar tvf randolph_carter.tgz` doit lister les fichiers et donner leur taille).

- **Une archive qui ne contient pas les fichiers demandés ne sera pas excusable.**
- ****Une archive qui n'est pas au format demandé** (archive tar gzippée avec suffixe .tgz) **ne sera pas corrigée**** donc c'est 0/20.

Toute tentative de fraude (plagiat, etc.) sera sanctionnée. Si plusieurs projets ont **des sources trop similaires** (y compris sur une partie du code uniquement), *tous* leurs auteurs se verront attribuer la note 0/20. En particulier, il faudra prendre soin de ne pas publier son travail sur un dépôt public (en tout cas pas avant la date de fin de rendu). On évitera également de demander (ou de donner) des conseils trop précis à ses camarades (y compris des promotions précédentes), ces conseils ayant pu être donnés à plusieurs personnes. Les rendus seront comparés deux à deux.

Procédure de dépôt
Vous devez enregistrer votre archive tgz dans le dépôt dédié au cours IPI (ipi-2020) en vous connectant à http://exam.ensiie.fr. Ce dépôt sera ouvert jusqu'au 4 janvier inclus.

Contexte

Le but de ce projet est d'implanter une plateforme de jeu de plateau.

Ce jeu se présente sous la forme d'un plateau de taille $n \times n$ dans lequel sont placés des piles de jetons de couleur, chaque joueur possédant une couleur différente. Initialement aucun jeton n'est présent.

Le jeu se joue à un nombre quelconque de joueurs qui jouent chacun à tour de rôle.

Le but du jeu est d'avoir le plus possible de jetons de sa couleur en haut des piles une fois que le plateau est complètement rempli.

Fonctionnement

- Initialement, le plateau est vide.
- Chaque joueur joue à tour de rôle.
- À chacun de ses tours, le joueur pose un jeton de sa couleur sur une des cases du plateau, en l'empilant sur les jetons déjà présents le cas échéant.
- Si les deux jetons au sommet de la pile sont de la même couleur, la case est activée :
  - les deux jetons sont retirés de la pile ;
  - des jetons de la couleur du joueur sont ajoutés sur les quatre cases voisines, à condition qu'elles n'aient pas déjà été activées à ce tour du joueur.
  - Les jetons ajoutés sur les cases voisines peuvent activer celles-ci, si elles avaient aussi un jeton de cette couleur en haut. Des jetons sont alors ajoutés sur les cases de leurs voisines qui n'ont pas déjà été activées ce tour, et ainsi de suite.

  Par exemple si les jetons au sommet du plateau sont les suivants :

```
 |1 2 3 4
-+-------
1|   2 2
2|   1
3|2 1 1 1
4|2   2 1
```

  si le joueur 1 place un jeton sur la case à la ligne 3 et la colonne 2, alors cette case est activée. On retire donc les deux 1 sur cette case, et on place des jetons 1 en (3,1); (2,2); (3,3) et (4,2). Les cases (2,2) et (3, 3) sont activées. On place des jetons sur (2,1); (1,2) et (2,3); ainsi que sur (2,3); (3,4) et (4,3). ((3,2) ayant déjà été activée, on ne place pas de jetons dessus.) Les cases (2,3) et (3,4) sont activées. On place des jetons sur (1,3) et (2,4); ainsi que sur (2,4) et (4,4). Les cases (2,4) et (4,4) sont activées. On place un jeton en (1,4); ainsi qu'en (4,3). Cette dernière case est activée. On place un jeton en (4,2). Cette case est activée. On place un jeton en (4,1). En supposant qu'il y avait des 3 sous les 1 au départ, on obtient le plateau suivant :

```
 |1 2 3 4
-+-------
1|   1 1 1
2|1 3
```

```
3|1 3 3 3
4|1   2 3
```

- Le but du jeu est d'avoir le plus possible de jetons de sa couleur au-dessus des cases, une fois que le plateau est rempli.

Interface

On travaille dans un premier temps en mode texte.

Au début de la partie, on demande le nombre *j* de joueurs, puis la taille du plateau *n*. Chaque joueur est identifié par un entier entre 1 et *j* qui est sa couleur.

À chaque tour du joueur *i*,

- On affiche le plateau vu du dessus.
- On demande au joueur d'entrer une coordonnée.
- On affiche la pile à la case correspondante.
- On demande confirmation au joueur pour déposer un jeton sur cette case. Si la réponse commence par `n`, on recommence à afficher le plateau, à demander une coordonnée, etc. Si la réponse commence par `o`, on place le jeton et on active les cases si approprié.

Si le plateau est rempli, c'est-à-dire qu'il y a au moins un jeton sur chaque case, alors on compte le nombre de jetons au sommet des cases pour chaque joueur et c'est celui qui en a le plus qui gagne.

Exemple d'affichage attendu ($n = 5$) :

```
Tour du joueur 1.

 |1 2 3 4 5
-+---------
1|  1 1 1
2|1 3     1
3|1 3 3 3
4|1   2 3 2
5|  2 1 3

Quelle case choisissez-vous ?
5 2

 2
 3
---

Voulez-vous mettre le jeton ici ?
n

Tour du joueur 1.

 |1 2 3 4 5
-+---------
1|  1 1 1
2|1 3     1
3|1 3 3 3
4|1   2 3 2
5|  2 1 3

Quelle case choisissez-vous ?
3 2

 3
 1
 2
 1
```

```
---

Voulez-vous mettre le jeton ici ?
o

Tour du joueur 2.

 |1 2 3 4 5
-+---------
1|  1 1 1
2|1 3     1
3|1 1 3 3
4|1   2 3 2
5|  2 1 3
```

Votre programme lira les commandes sur l'entrée standard. En particulier, il devra fonctionner correctement en lisant le fichier suivant `partie.txt` en entrant

```
./mon_programme < partie.txt
```

Conseils

Pour la récupération d'une entrée de l'utilisateur, plutôt que faire un `scanf` directement, il vaut mieux récupérer une ligne en entier avec `fgets` puis utiliser `sscanf` dessus ;on peut utiliser la suite de commandes suivantes :

```
char buf[256];
 .
 .
 .
fgets(buf, 256, stdin);
sscanf(buf, "format", ...);
```

Vous devez avoir lu jusqu'ici avant de commencer.