heb
esi

Nom : .....
Prénom : .....
Groupe : .....
Identifiant : .....

/ 50

Haute École de Bruxelles
École Supérieure d'Informatique
Bachelor en Informatique

2014 – 2015

Développement – 1ère

Examen de Janvier

Serpents et échelles

Jeudi 15 janvier 2015

Consignes

L'examen dure 4h. Gérez bien le temps afin de pouvoir répondre à l'ensemble des questions (algorithmique et langage Java)

Serpents et échelles est un jeu de société populaire, se jouant à plusieurs, consistant à déplacer les jetons sur un tableau de cases avec un dé en essayant de monter les échelles et en évitant de trébucher sur les serpents.

Pour l'histoire, le jeu peut être perçu comme une représentation d'un chemin spirituel que les humains prennent pour atteindre le ciel. Avec des bons gestes, le chemin est raccourci (ce que symbolisent les échelles), tandis qu'avec de mauvais gestes, le chemin est allongé (d'où vient le symbolisme des serpents).

Extrait de Wikipedia.

Version du document : 14 janvier 2015

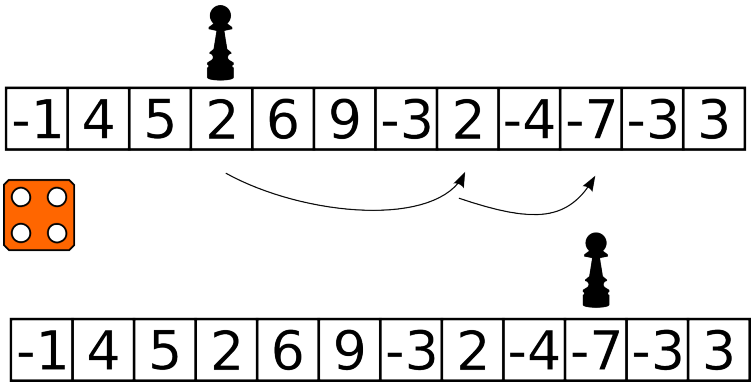
Représentation du chemin. Nous représenterons le chemin par un tableau d'entiers. Chaque case du tableau contiendra une valeur :
– positive pour représenter une échelle et permettre d'avancer plus vite ;
– négative pour représenter le serpent et ralentir (voire reculer).

Règles du jeu. Pour jouer, le joueur qui a la main lance le dé et avance de la valeur indiquée par le dé. À partir de cette nouvelle position, il vérifie la valeur de la case sur laquelle il se trouve pour continuer à avancer ou au contraire reculer. Si la valeur est positive, il avance et si elle est négative, il recule de la valeur contenue dans la case.

Si la case est déjà occupée, le joueur se placera sur la case suivante.

Le joueur passe la main au joueur suivant.

Exemple. Le joueur se trouvant à la position 3 lance le dé. Il obtient la valeur 4. Il avance de 4 cases et se retrouve sur la case en position 7 qui contient la valeur 2. Il avance encore de 2 cases ... et termine donc son tour en position 9. Il passe alors la main au joueur suivant.



Le premier joueur à atteindre ou dépasser la dernière case a gagné. Nous supposerons que cette dernière case est en position 42. Nous supposons également que la première et la dernière case n'ont ni échelle ni serpent (valeur nulle).

Positions des joueurs. Les positions des joueurs seront mémorisées dans un tableau d'entiers. Toutes les cases de ce tableau sont initialisées à 0.

Le plateau de jeu, un tableau d'entiers s'appellera **chemin**.

Le tableau d'entiers contenant la position courante des joueurs s'appellera **positionsJoueurs**.

S'il y a 4 joueurs, le tableau *positionsJoueurs* contient [2,5,1,7] alors :
– le joueur 0 est en position 2 sur le chemin ;
– le joueur 1 est en position 5 sur le chemin ;
– le joueur 2 est en position 1 sur le chemin ;
– le joueur 3 est en position 7 sur le chemin ;

I
Algorithmique

Consignes

Pour la partie algorithmique,
- Vous travaillez sur papier, vous ne pouvez pas utiliser de notes ni d'ordinateur.
- Vos réponses se feront au bic bleu ou noir sur la feuille de réponses.
- Sauf spécification du contraire, les données lues ou reçues ne comportent pas d'erreurs.
- Les noms en gras (variables et types) doivent être respectés.
- Veillez à travailler de manière modulaire.

1 Le premier jour, initialiser le chemin (4 points)

Écrivez un module

module *créerChemin*(n↓ : entier) → **tableau** [1 à n] d'entiers

Ce module *créerChemin* créera un tableau d'entiers dont les valeurs seront des valeurs aléatoires comprises strictement entre -10 et 10 (inclus).

Vous pouvez supposer l'existence d'un module `hasard(n)` retournant un nombre (entier) aléatoire compris entre 1 et n (inclus).

Le tableau retourné aura une taille de n.

Nous supposons que l'entier n est un naturel strictement supérieur à 0. Vous ne devez pas le vérifier.

2 Qui est en tête ? (5 points)

Écrivez un module

module *enPremièrePosition*(positionsJoueurs↓ : **tableau** [1 à n] d'entiers)
→ entier

Ce module *enPremièrePosition* retourne le numéro du joueur en tête de la course. Il recherche donc le maximum dans le tableau passé en argument et retourne l'indice de ce maximum.

3 Jouer un tour de jeu (8 points)

Ce module permettra de jouer un tour de jeu pour un joueur donné. Jouer un tour de jeu consiste à le faire avancer ou reculer (c'est à dire changer sa position) du bon nombre de cases.

Écrivez un module

module *jouer*(chemin↓ : **tableau** [1 à n] d'entiers,
positionsJoueurs↓↑ : **tableau** [1 à m] d'entiers, joueurCourrant↓ : entier,
valeurDé↓ : entier)

Ce module *jouer* fait changer de position le joueur d'indice *joueurCourrant* de la valeur donnée par le dé (*valeurDé*) en respectant les règles du jeu.

Ce module met donc à jour le tableau *positionsJoueurs*.

Comme le joueur ne peut se placer sur une case déjà occupée, il peut être utile d'écrire un module `estOccupé` précisant si la case est libre ou occupée.

4 Classement (8 points)

Soit le tableau *positionsJoueurs* (indicé à partir de 0) donnant la position de chaque joueur. Par exemple, dans le tableau `[6,11,9,7]`, le joueur 0 est en position 6, le joueur 1 en position 11, le joueur 2 en position 9 et le joueur 3 en position 7.

Écrivez un module

module *indiceMaxPlafonné*(`tab`↓ : **tableau** [0 à n] d'entiers, `plafond`↓ : entier)

qui retourne l'indice de la plus grande valeur de `tab` strictement inférieure à `plafond`. Dans le cas où aucune valeur n'est trouvée, on retourne -1. Nous supposons que le tableau ne contient pas de doublons.

Exemples

```
indiceMaxPlafonné([6,11,9,7], 50) -> 1
indiceMaxPlafonné([6,11,9,7], 10) -> 2
indiceMaxPlafonné([6,11,9,7], 9)  -> 3
indiceMaxPlafonné([6,11,9,7], 3)  -> -1
```

II
Java et laboratoire

Consignes

Pour la partie java,

- *Vous réaliserez votre travail sur* `linux1` *et le déposerez dans le casier linux de votre professeur par la commande* `casier`.
- *Vous disposez de toutes vos notes ainsi que de l'aide en ligne.*
- *Il ne suffit pas que votre code compile. Testez-le pour identifier d'éventuelles erreurs à l'exécution.*
- *La cotation tiendra compte aussi du style de programmation que vous avez acquis.*
- *Respectez bien les noms de package, classe, méthodes demandés dans l'énoncé.*
- *Vous remplacerez bien sûr g12345 par votre numéro d'étudiant.*

5 Question préalable (0 point)

Créez un répertoire `evaluations/janvier`. Changez les droits sur votre répertoire `janvier` pour donner les permissions de lecture et d'exécution aux professeurs mais aucun droit aux autres étudiants. Appelez votre professeur pour lui montrer que vos permissions ont bien été changées.

Vous ne continuerez pas l'examen tant que cette question n'a pas été validée par votre professeur.

6 Mise en place (3 points)

Dans la suite, votre classe s'appellera `g12345.evaluation.janvier.SerpentsEchelles`.

Écrivez ici (l'important dans les questions qui suivent est la cohérence de l'ensemble) :

- votre répertoire de travail (probablement `~/evaluations/<votre choix>`) ;

- l'instruction que doit contenir votre classe pour faire partie du package demandé ;

- la commande (complète et précise) que vous utilisez (à partir de votre répertoire courant) pour **compiler** votre classe ;

– la commande (complète et précise) que vous utilisez (à partir de votre répertoire courant) pour **exécuter** votre classe ;

– le contenu minimal de votre variable d'environnement `CLASSPATH`

7 Recherche de l'indice maximum plus petit que ... (6 points)

Écrivez la méthode `indiceMaxPlafonné` comme décrite dans la partie algorithmique.

8 Tests JUnit (6 points)

Écrivez au minimum 5 tests JUnit permettant de tester la validité de la méthode `indiceMaxPlafonné`.

Ces tests se trouveront dans une classe `TestSerpentsEchelles`.

Écrivez ici :

– la commande permettant de lancer les tests JUnit [1] ;

– le contenu de la variable d'environnement `CLASSPATH` pour pouvoir exécuter ces tests ;

– une commande permettant de **rediriger** les résultats des tests dans le fichier `tests.log`.

1. Inutile d'écrire ici un `alias`

9 Les autres méthodes (6 points)

Écrivez les méthodes :

- `public static int[] créerChemin(int n);`
- `public static int enPremièrePosition(int[] positionsJoueurs);`
- `public static void jouer(int[] chemin, int[] positionsJoueurs, int joueurCourrant, int valeurDé)`

Écrivez une méthode permettant de donner le classement des joueurs. Par exemple si le tableau *positionsJoueurs* vaut `[3,4,1,7]`, alors le classement des joueurs est le suivant :

1. joueur 3 (en position 7)
2. joueur 1 (en position 4)
3. joueur 0 (en position 3)
4. joueur 2 (en position 1)

Cette méthode se base sur la méthode `indiceMaxPlafonné`.

Vous n'avez pas écrit de méthode permettant de donner le classement dans la première partie. Cette méthode fait des appels successifs à la méthode `indiceMaximumSousPlafond`.

- `public static void afficherClassement(int[] positionsJoueurs)`

10 Javadoc (4 points)

Si vous ne l'avez pas fait au fur et à mesure, écrire la **javadoc** pour vos méthodes publiques et la générer dans un sous-répertoire `doc`.