Université Paris 7 – Denis Diderot
Introduction à l'informatique et à la programmation IF
Examen du 6 janvier 9 Durée heures
Documents, calculettes, ordinateurs et téléphones non autorisés

---

Les exercices sont indépendants et peuvent être traités dans n'importe quel ordre. Dans chaque exercice, on pourra traiter une question en supposant que les précédentes l'ont été et donc utiliser les fonctions correspondantes (si cela est utile).

Exercice 1.

On considère la définition suivante d'une fonction :

```
public static char fonc(boolean a, boolean b, boolean c) {
  if (a || !b)
       if (c) return ('A');
       else if(!a && c) return('B');
            else return('C');
  else if (!c && (a || b)) return('D');
       else if (b) return('E');
            else return('F');
}
```

1.1. Quelle est la valeur de la fonction `fonc` pour chacune des combinaisons possibles de valeurs des variables a, b et c ?

1.2. Donner une définition équivalente de cette fonction sans tests imbriqués, c'est-à-dire exprimée sous la forme `if ... else if ... else if ... ... else`

Exercice 2.

L'objet de cet exercice est de déterminer le nombre d'entiers différents contenus dans un tableau d'entiers donné.

2.1. On souhaite tout d'abord ne compter que les entiers compris entre 0 et 20 (bornes incluses) apparaissant dans le tableau, les autres n'étant pas comptabilisés.
Écrire une fonction qui, étant donné un tableau d'entiers `t`, retourne le nombre d'entiers compris entre 0 et 20 apparaissant au moins une fois dans le tableau `t`.
Ainsi, appelée avec le tableau `[1 2 -1 11 2 1 2 23 11 3 1 -1 0 9 1 11]`, la fonction renverra 6. En effet, il y a six nombres compris entre 0 et 20 dans le tableau (0, 1, 2, 3, 9 et 11).

Indication : on pourra utiliser un tableau de booléens (de taille définie par l'intervalle) permettant de mémoriser les nombres rencontrés dans le tableau et un compteur pour compter les entiers différents (égal au nombre de booléns du tableau égaux à `true`).

2.2. On souhaite maintenant compter **tous** les nombres rencontrés au moins une fois dans le tableau sans fixer d'intervalle de définition pour ces nombres.
Écrire une fonction qui, étant donné un tableau d'entiers `t`, retourne le nombre d'entiers différents apparaissant au moins une fois dans le tableau `t`.
Appelée avec le tableau `[1000 -10 11 2 1000 2 -230000 11 3 110000 -10 1000 2]`, la fonction renvoie 7 (le tableau contient sept nombres différents : -230000, -10, 2, 3, 11, 1000 et 110000).

Indication : ne pouvant procéder comme dans le cas précédent, pour chaque élément du tableau `t[i]`, on vérifiera s'il est ou non égal à un `t[j]` avec `j<i` afin de décider s'il doit être rejeté ou retenu.

Exercice 3. *Jeu du Mastermind (à partir de 6 ans)*

Le jeu est un plateau de 12 rangées de 4 trous destinés à recevoir des pions de couleur. Il y a des pions de 8 couleurs désignées dans l'exercice par les caractères `'A'`, `'B'`, `'C'`, `'D'`, `'E'`, `'F'`, `'G'` et `'H'`.
Le premier joueur choisit une combinaison de 4 pions qu'il garde secrète : chacun des pions peut être de l'une quelconque des 8 couleurs (et plusieurs pions peuvent être de même couleur).

Le jeu consiste en une suite de rondes (au plus 12) visant à permettre au second joueur de retrouver la combinaison secrète. Chaque ronde comporte :

1. une phase de proposition : le second joueur propose une combinaison de 4 pions ;
2. une phase de réponse : le premier joueur (qui connaît la solution puisqu'il l'a choisie) indique :
   (a) le nombre de pions bien placés dans la proposition (couleur correcte, position correcte) ;
   (b) parmi les autres pions (ceux non pris en compte dans la phase (a)), le nombre de pions mal placés (couleur correcte, position incorrecte).

Par exemple, si le premier joueur a choisi la combinaison `CEBF`, des exemples de propositions et de réponses sont :

- `ABCD` : 2 mal placés (le `B` et le `C`), ce qu'on traduira par `--` ;
- `BCBE` : 1 bien placé (le second `B`) et 2 mal placés (le `C` et le `E`), ce qu'on traduira par `+--`.

Une réponse est donc une suite de m caractères `+` suivie d'une suite de n caractères `-` où m est le nombres de pions bien placés et n le nombre de pions mal placés. La fonction renverra `null` si m et n sont nuls tous les deux.

3.1. Écrire une fonction `trouve` qui, étant donnés deux tableaux de caractères `tab1` (la proposition) et `tab2` (la solution) retourne un tableau de deux entiers dont le premier est le nombre de pions bien placés et le second est le nombre de pions mal placés.
On veillera à ce qu'un pion de la proposition ou de la solution ne serve qu'une fois et prioritairement pour indiquer un pion en bonne position dans la solution.
La fonction sera utilisable pour des tableaux de taille quelconque, mais renverra `null` si les deux tableaux n'ont pas la même taille.

3.2. Écrire une fonction `reponse` qui, étant donné un tableau `t` de deux entiers, retourne un tableau contenant `t[0]` caractères `+` suivis de `t[1]` caractères `-`. Vous pouvez, si vous le préférez, renvoyer une chaîne de caractères plutôt qu'un tableau.

3.3. Écrire un programme qui simule une partie contre l'ordinateur, c'est-à-dire réalise la séquence suivante :

1. choix d'une combinaison de 4 pions : on supposera qu'il existe une fonction `char couleur()` qui retourne l'un des 8 caractères associés à une couleur (la valeur renvoyée est supposée choisie aléatoirement) ;
2. possibilité pour l'utilisateur de faire un maximum de 12 propositions : le programme répondra entre chaque proposition sous la forme d'un mot formé de caractères `+` tel que décrit en 3.2 ;
3. arrêt du programme soit si l'utilisateur a trouvé la bonne combinaison en le félicitant, soit au bout de 12 propositions en dévoilant la solution ;
4. mémorisation de toutes les propositions (pour permettre un contrôle éventuel en fin de partie).

Exercice 4.

Toute suite finie d'entiers peut être décomposée de manière unique en une suite de séquences strictement croissantes maximales. En représentant une suite dans un tableau `t`, le tableau

`[`1 2 5 7 2 6 0 5 2 4 6 7 8 9 3 4 6 1 2 7 8 9 4 2 3 1 5 9 7 1 6 6 3`]`

se décompose ainsi en le tableau de 13 tableaux d'entiers suivant :

`[[1 2 5 7][2 6][0 5][2 4 6 7 8 9][3 4 6][1 2 7 8 9][4][2 3][1 5 9][7][1 6][6][3]]`.

Les premiers éléments de ces séquences sont, en plus du 1-er élément du tableau, les éléments (soulignés sur l'exemple) qui sont inférieurs ou égaux à leur prédécesseur dans le tableau (`t[i]`$\leq$`t[i-1]`).

4.1. Écrire une fonction `rupture` qui, étant donné un tableau `t` d'entiers, retourne un tableau contenant `0` en premier élément et les indices des éléments de `t` inférieurs ou égaux à leur prédécesseur en ordre croissant. Pour le tableau donné en exemple, la fonction retourne le tableau :

`[0 4 6 8 14 17 22 23 25 28 29 31 32]`.

4.2. Écrire une fonction `factorisation` qui, étant donné un tableau `t` d'entiers, renvoie un tableau bidimensionnel d'entiers, c'est-à-dire un tableau de tableaux d'entiers, dont la i-ème ligne contient la i-ème plus longue séquence croissante de nombres adjacents dans le tableau `t` (résultat tel que celui donné dans l'exemple).

Indication : on pourra utiliser 4.1 pour determiner le nombre de lignes et la taille de chaque ligne de ce tableau bidimensionnel.

Exercice 5/Problème. *Le jeu Othello/Reversi (à partir de 8 ans)*
Ce jeu se joue à deux, sur une table carrée de 64 ca54.331(O)697ei un52(`)5553ce8