IN 101 - Examen
11 janvier 2022
François Pessaux

Énoncé

— **Lisez attentivement les consignes et tout le sujet avant de commencer.**

— Les documents (polys, transparents, TDs, livres ...) sont autorisés.
— Sont **absolument interdits** : le WEB, le courrier électronique, les messageries diverses et variées, le répertoire des camarades, le téléphone (même pour avoir l'heure puisque vous l'avez sur votre ordinateur).

— Votre travail sera (en partie) évalué par un mécanisme automatique. Vous **devez** respecter les règles de **nommage** des fichiers et autres **consignes** qui vous sont données.
— La connaissance de C est un pré-requis d'IN101. Ainsi, la présence d'avertissement(s) **réduit la note de 2 pts** (sur 20) de l'exercice considéré, un programme ne compilant pas aura une note **bornée à 10 pts**.
— **Sauf indications contraires**, vos programmes doivent **gérer** les cas **d'erreur** pouvant survenir.
— Lorsqu'il vous est demandé que votre programme réponde en affichant « **Yes** » ou « **No** », il ne doit **rien** afficher d'autre, et **pas** « Oui » ou « Yes. » ou « no » ou « La réponse est : no ». Donc, pensez à retirer vos affichages de test / debug.
— La **lisibilité** et l'efficacité / simplicité / complexité de vos programmes seront **prises en compte** dans l'évaluation.

— **À la fin de l'examen**, vous devrez créer une **archive** contenant **tous** les fichiers **sources** que vous avez écrits (`.c`, `.h`). **N'incluez pas** d'exécutables dans l'archive, le mail pourrait la considérer comme un attachement dangereux. Le nom de cette archive devra avoir la structure suivante :
`nom_prenom.zip` ou `.tgz` (selon l'outil d'archivage que vous utilisez).
— Vous devrez **m'envoyer** cette archive par mail. En cas d'envoi incorrect, il vous sera demandé de refaire l'envoi. Par contre, vous ne devrez **surtout pas modifier** les fichiers : leur date de dernière modification ne devra pas être ultérieure à l'heure de fin de l'épreuve sous peine d'être considérés comme nuls.
— **N'oubliez pas** d'effectuer cet envoi sinon nous devrons considérer que vous n'avez rien rendu !

— Le sujet comporte 5 pages et l'examen dure **2h30**.
— Le barème est **volontairement** approximatif.

1 Jeu de dames bizarre ($\sim 60\%$)

Nous considérons un jeu de dames avec des règles légèrement modifiées. Le jeu se joue sur un damier carré de taille réglable au début de la « partie ». Le but est de « faire une dame », donc d'amener un pion à l'opposé de sa zone de départ.

On ne s'intéressera qu'à la situation du joueur du « **haut** » du damier, qui doit donc amener ses pions en « bas » du damier.

Un pion ne peut se déplacer **qu'en sautant par-dessus** un autre qui lui est immédiatement voisin, **sans jamais remonter**. Néanmoins, contrairement aux dames classiques, il peut se déplacer **latéralement**. En un tour, un pion peut se déplacer un nombre quelconque de fois tant qu'il a la possibilité de sauter par-dessus un autre. Contrairement au jeu normal, les pions sautés **restent** sur le damier.

But macroscopique du programme : On souhaite réaliser un programme qui vérifie si un pion dont la position sera demandée à l'utilisateur peut, selon une configuration du damier obtenue via un fichier texte, « devenir une dame » **en un tour** et si oui en **combien de sauts au minimum**. Autrement, est-ce qu'il peut effectuer une succession de sauts, et combien, de sa position initiale jusqu'en bas du damier. Notez que la position n'est pas forcément sur la première ligne.

Exemples de configurations :

```
dames.c       Mon Jan 10 09:21:19 2022       1
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>


/* Nombre max de sauts en fonction de la largeur du damier. */
#define MAX_NB_JUMPS(size) ((((size) - 1) / 2) * (((size) - 1) / 2) + (((size) - 1) / 2))
/* En global pour se simplifier la vie. Distance plus longue que la longueur
   maximale d'une chaÃ®ne de sauts sur le damier. */
int impossible_jumps = 0 ;


/* D'oÃ¹ on venait lors du dernier saut qui nous a amenÃ© sur une case. Sert
   Ã  Ã©viter de boucler en faisant droite <-> gauche indÃ©finiment entre 2
   mÃªmes cases. */
enum prev_mov_t { Left, Right, Bottom } ;


int min (int a, int b) { if (a < b) return a ; return b ; }

void cleanup_grid (char **grid, int size) {
  if (grid) {
    for (int y = 0; y < size; y++) {
      if (grid[y] != NULL) free (grid[y]) ;
    }
    free (grid) ;
  }
}

void cleanup_dists (int **dists, int size) {
  if (dists) {
    for (int y = 0; y < size; y++) {
      if (dists[y] != NULL) free (dists[y]) ;
    }
    free (dists) ;
  }
}


/* Copie le contenu de tmp_buff dans grid[y]. Retourne un boolÃ©en disant si la
   copie a Ã©tÃ© un succÃ¨s (longueur de tmp_buff Ã©gale Ã  size). */
bool copy_line (char **grid, char *tmp_buff, int y, int size) {
  int x ;
  int llen = strlen (tmp_buff) ;
  if (llen != size) return false ;
  for (x = 0; x < llen; x++) grid[y][x] = tmp_buff[x] ;
  return true ;
}


/* Fonction de chargement du damier. Retourne NULL en cas d'erreur. */
char** load_game (char *fname, int *size_out) {
  int size ;
  char **grid ;
  int y ;
  char tmp_buff[256] ;
  FILE *file = fopen (fname, "rb") ;
  if (file == NULL) return NULL ;

  fscanf (file, "%d", &size) ;
  if (size <= 0) {
    fclose (file) ;
    return NULL ;
  }

  grid = malloc (size * sizeof (char*)) ;
  for (y = 0; y < size; y++) grid[y] = malloc (size * sizeof (char)) ;
```

Dans le damier de gauche, le pion de position $(x, y) = (0, 0)$ peut effectivement arriver en bas du damier en 4 sauts. Dans le damier du milieu il ne peut pas car le pion en position $(x, y) = (4, 2)$ bloque le saut. Dans le damier de droite, il peut également arriver en bas du

damier en se déplaçant latéralement d'abord vers la droite puis en revenant vers la gauche.

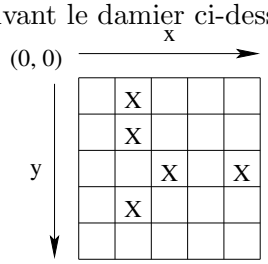
Description de la configuration du damier : La configuration du damier sera obtenue depuis un fichier texte contenant :

1. la largeur du damier (que l'on appellera *size*),
2. *size* lignes composées des caractères :
   — `'X'` : pour représenter un pion
   — `'.'` : pour représenter une case vide

Exemple : `game1.txt`

```
5
.X...
.X...
..X.X
.X...
.....
```

Ce fichier est la représentation des données décrivant le damier ci-dessous.



Nommage : Le fichier source de ce programme devra s'appeler `dames.c`.

Format d'entrée : Votre programme prendra en arguments de **ligne de commande**, et **dans cet ordre** : le nom du fichier de damier, la position x, la position y du pion que l'on veut tester.

Format de sortie :

— S'il est **possible** de faire une dame, la **dernière ligne** d'affichage devra être : « `Dame en` x `coups.` » ou x sera le nombre de sauts calculé. La ligne sera terminée par un **retour à la ligne**.
— S'il est **impossible** de faire une dame,la **dernière ligne** d'affichage devra être : « `Dame impossible.` » suivi d'un **retour à la ligne** final.

Les éventuels messages d'erreur sont libres.

Jeux de données : Des jeux de tests vous sont donnés à titre d'exemples :

— `game1.txt` : position $(1, 0)$ → Dame en 2 coups.
— `game2.txt` : position $(0, 0)$ → Dame en 6 coups.
— `game2.txt` : position $(6, 1)$ → Dame impossible.
— `game3.txt` : position $(0, 0)$ → Dame en 3 coups.
— `game3.txt` : position $(0, 3)$ → Dame impossible.
— `game4.txt` : position $(6, 0)$ → Dame en 3 coups.
— `game5.txt` : position $(5, 0)$ → Dame en 8 coups.
— `game6.txt` : position $(5, 0)$ → Dame en 11 coups.

Réponses aux questions de réflexion intermédiaires : Elles devront apparaître en commentaire en début de votre fichier source. Si vous souhaitez transmettre des photos de schémas, de brouillons, vous serez autorisés à nous envoyer après l'examen ces images, dans la limite **d'une heure après la fin de l'épreuve**. Cela vous laissera le temps d'extraire les photos de votre téléphone et de les envoyer par mail. Merci de ne faire qu'un seul mail pour ces potentielles images.

1.1 Q1.1

Identifiez les grandes étapes du programme, sans descendre dans le détail des fonctions que vous allez écrire. Cela représentera en quelque sorte votre futur `main`.

1.2 Q1.2

Identifiez les grandes étapes de la fonction de chargement du damier. Quels sont ses domaines d'entrée et de sortie ? La gestion des cas de fichier mal formé sera un bonus.

1.3 Q1.3

Quel est le nombre maximal de sauts en fonction de la largeur du damier ? Si vous n'arrivez pas à répondre à cette question, ne perdez pas tout votre temps, vous utiliserez alors une valeur arbitraire (assez grande). Réfléchissez à quoi ce nombre va vous servir dans l'algorithme d'exploration.

1.4 Q1.4

Identifiez de quelle manière vous allez explorer le damier. On vous rappelle que l'on n'a pas le droit de remonter. Attention, pensez au cas gênant qui consisterait à sauter d'un côté puis immédiatement après de l'autre coté... à l'infini.

1.5 Q1.5

Si ce n'est déjà fait, réfléchissez à comment vous pourriez gagner du temps durant l'exploration. Par exemple, considérez le damier ci-contre du fichier `game6.txt`. En partant de la position $(x, y) = (5, 0)$, quel(s) chemin(s) pouvez-vous emprunter ? Que remarquez-vous à propos de la fin du/des chemin(s) ? Qu'avez-vous envie de faire pour gagner du temps ?

Remarque : si vous ne trouvez pas la réponse à cette question et que votre algorithme semble fonctionner, ne perdez pas tout votre temps, vous y reviendrez éventuellement plus tard s'il vous reste du temps.

```
11
..X.XXX.X..
.X.......X.
...........
.X.......X.
..X.X.X.X..
.......X...
...........
.......X...
..X.X.X...X
.X.......X.
....X......
```

1.6 Q1.6

Transformez votre réflexion algorithmique en un programme effectif. Respectez les formats d'entrée et de sortie spécifiés en introduction de cet exercice.

2 Chiffres décroissants ($\sim 30\%$)

On souhaite déterminer si un *entier* est composé de chiffres décroissants **de 1 à chaque fois** lorsque l'on lit son écriture en **base 10** de **gauche à droite**. Si tel est le cas alors on la vérification devra «dire » `Ok`, sinon `Ko`.

Exemples : 543 a bien des chiffres décroissants car il est formé de 5, 4, 3, chacun étant le précédent - 1 (sauf l'initial 5 bien entendu).

Par contre, ce n'est pas le cas pour 532 car il est composé de 5, 3, 2 or 3 n'est pas le prédécesseur immédiat de 5.

Vous écrirez une fonction `dec` qui reçoit cet entier, effectue la vérification et **retourne** son verdict. Vous écrirez un `main` qui se charge d'appeler votre fonction et d'afficher le résultat.

Votre programme prendra l'entier en **argument de ligne de commande**.

Nommage : Le fichier source de ce programme devra s'appeler `dec.c`.

Format de sortie : **uniquement** la chaîne « `Ok` » si le test est positif, « `Ko` » s'il est négatif, le tout terminé par un **retour à la ligne**. Les éventuels messages d'erreur sont libres.

Ex. tests :

- `./dec.x 345` ⟶ `Ko`↵
- `./dec.x 432` ⟶ `Ok`↵

Votre fichier source débutera, en commentaire, par une **description du principe de votre algorithme** en quelques lignes (synthétisez, court mais clair). Celle-ci devra rendre compte de votre analyse du problème, des raisons qui vous ont conduit à votre solution. Cet élément explicatif entrera dans la note de l'exercice : plus vite je comprendrai votre algorithme à la lueur de cette description, meilleure sera la note donnée à cet élément explicatif. Même remarque à propos d'éventuelles photos de brouillons que pour l'exercice 1.

—— Fin du sujet ——