DLST

UNIVERSITÉ Grenoble Alpes

INF203 - Examen terminal
Année 2024-2025

Durée : 2h.
Une feuille A4 manuscrite recto/verso autorisée, dictionnaire papier (non annoté) autorisé pour les étudiants étrangers uniquement.
Tout autre document, calculatrices et appareils électroniques interdits.
Pour chaque question, une partie des points peut tenir compte de la présentation.
Le barème est indicatif.

Toute réponse même incomplète sera valorisée à partir du moment où elle réalise au moins une partie de ce qui est demandé. Les questions sont relativement indépendantes, et dans tous les cas vous pouvez utiliser les scripts et les fonctions demandées dans les questions précédentes même si vous n'avez pas réussi à les écrire.

Par ailleurs, tout code ou algorithme *élégant*, en plus d'être correct, sera bonifié.

1 Langages C et Bash : Classement (15 points)

Dans cette partie, on souhaite entrer les notes des élèves de la promo et les classer par ordre croissant. On se donne la structure suivante :

```
struct eleve {
    char nom[20];
    char groupe[5];
    double note;
};
```

1. (2 points) Écrire une fonction `verif_eleve` de profil

   ```
   int verif_eleve (struct eleve * p)
   ```

   qui renvoie 1 (vrai) si la fiche de l'élève (passée en argument par adresse) vérifie les critères suivants :
   - le nom est composé uniquement de lettres ou de tirets ;
   - la note est comprise entre 0 et 20

   et renvoie 0 (faux) sinon.

2. (2 points) Écrire une fonction `lire_eleve` de profil

   ```
   void lire_eleve (struct eleve * p)
   ```

   qui lit depuis l'entrée standard les différents champs correspondant à un·e élève et qui les stocke dans une structure `eleve` passée en argument par adresse.

   On suppose que le nom, le groupe et la note de l'élève sont donnés dans cet ordre sur l'entrée standard, séparés par des espaces.

3. (2 points) Supposons que nous avons accès au tableau `struct eleve T[200]` dans lequel ont été stocké·es `n` élèves (`n` est donc inférieur à 200). Écrire une fonction `classement` de profil

   ```
   void classement (int n, struct eleve T[])
   ```

   qui prend en argument un nombre d'éléments `n` à lire dans un tableau `T` et qui réorganise le tableau `T` dans l'ordre croissant des notes des élèves.

   Par exemple, si `n` vaut 2 et que

```
T[0]={"Anne","IMA5",13.5};
T[1]={"Bernard","INF2",10.0};
```

(et `T[i]` peut contenir n'importe quelle entrée pour `i` > 1), la fonction classement réorganisera `T` de telle manière que

```
T[0]={"Bernard","INF2",10.0};
T[1]={"Anne","IMA5",13.5};
```

et `T[i]` reste inchangé pour `i` > 1.

Indication : Pour réaliser le classement, on pourra par exemple boucler sur les `n` entrées de `T` et remplacer l'entrée d'indice courant (disons `T[i]`) par l'entrée `T[j]` dont la note est la plus petite parmi les `j` $\geq$ `i`.

4. (3 points) Écrire une fonction principale `main` de profil

   ```
   int main()
   ```

   qui demande à l'utilisateur un entier n, vérifie que n soit inférieur à 200 (et retourne une erreur sinon), puis qui lit les données de n structures `eleve` et affiche à l'écran ces n structures triées par notes croissantes.

5. (1 point) Toutes les fonctions ci-dessus définissent le programme `classement.c`. Quelle(s) librairie(s) doit-on inclure (avec la commande `#include`) pour que clang puisse le compiler ? Quelle est alors la commande de compilation permettant de générer l'exécutable `classement` ?

6. (2 points) Depuis sa console Bash, l'utilisateur ne souhaite pas systématiquement entrer le nombre d'élèves, puis l'ensemble des noms, groupes et notes depuis l'entrée standard (clavier) à chaque exécution de `classement`. Il préfèrerait stocker l'ensemble de ces données dans un fichier externe `eleves.txt`.

   Comment ce fichier doit-il être organisé ? Comment l'utiliser comme entrée de `classement` dans la console Bash ?

7. (3 points) Après réflexion, il paraît plus pertinent de directement choisir avec le programme `classement` si l'on souhaite lire les données élèves (noms, groupes, notes) depuis un fichier externe ou depuis l'entrée standard (clavier). Pour cela, on souhaite modifier le programme `classement` de sorte qu'il puisse prendre comme argument le nom du fichier en question et, si aucun aucun argument n'est donné, le programme lira depuis l'entrée standard (clavier) comme auparavant.

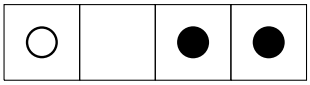
   Réécrivez les fonctions `main` et `lire_eleve` afin de permettre ce nouveau comportement. On retournera une erreur si l'utilisateur fait le choix de lire un fichier mais que ce fichier n'existe pas ou n'est pas accessible en lecture.

2 Automates (6 points)

Dans cette partie, vous devez dessiner un automate permettant de modéliser la règle du jeu suivant :

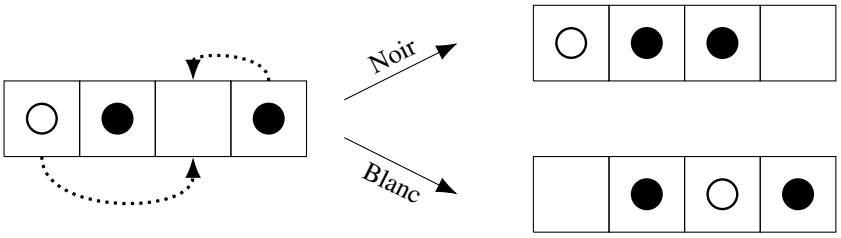
— On joue sur une ligne de 4 cases.

— Initialement la case la plus à gauche contient un pion blanc et les deux cases les plus à droite contiennent chacune un pion noir :

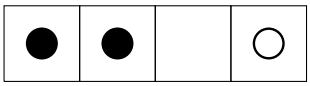


— Le pion blanc ne se déplace que vers la droite, et les pions noirs que vers la gauche.

— Quand il se déplace, un pion peut : soit avancer sur une case vide **voisine**, soit sauter par-dessus **un seul** pion de l'autre couleur pour atteindre une case vide. Par exemple :



— Le but du jeu est de placer les pions noirs dans les deux cases les plus à gauche, et le pion blanc dans la case la plus à droite :



Chaque état de l'automate sera une configuration du plateau de jeu. Les entrées sont $\{Blanc, Noir\}$ selon la couleur du pion qu'on décide de déplacer (on admettra qu'il n'y a jamais plus d'un pion noir déplaçable).

L'automate doit comporter un ou plusieurs états finaux, qui seront atteints lorsque la partie se termine. On ne demande pas d'écrire des transitions sortant des états finaux. On ne représentera pas non plus les transitions quand un coup n'est pas autorisé (par exemple déplacer le pion blanc alors qu'il est déjà tout à droite).

Il y aura trois sorties possibles $\{Gagné, Perdu, Rien\}$:

— les deux premières sorties sont associées aux transitions qui permettent d'atteindre un état final ;

— la sortie « Rien » est associée aux autres transitions et peut être omise sur vos dessins (sortie par défaut).

1. (4 points) Dessiner l'automate qui traduit cette règle de jeu.
2. (2 points) À l'aide de cet automate, répondre aux questions suivantes :
   — Quelles sont toutes les suites de coups perdantes ?
   — Combien y a-t-il de suites de coups gagnantes distinctes ?
   — En combien de coups peut-on parvenir au but du jeu, au minimum et au maximum ?