Examen de Méthodologie de conception et de programmation

Licence Informatique, 2016-2017
Durée : 2h
Documents interdits

Partie I. Le dilemme de Maya (/14)

Maya est polyamoureuse. C'est à dire qu'elle accepte l'idée que l'amour se multiplie plutôt qu'il ne se divise, et qu'il n'y a pas de raison de s'interdire d'aimer une personne si on en aime déjà une autre. À ce titre, elle entretient en toute honnêteté **plusieurs relations amoureuses**, avec plusieurs personnes (qui sont elles-mêmes éventuellement impliquées dans plusieurs relations). Pratiquer le polyamour dans une société qui a utopiquement pris pour norme l'exclusivité sentimale, implique parfois de résoudre quelques dilemmes au quotidien.

Pour l'anniversaire de son frère Cassy, sa famille lui a demandé de venir accompagnée d'une seule personne. Maya ayant fait le choix de ne pas hiérarchiser ses relations, **elle décide de faire voter les membres de sa famille** : ils devront décider eux-mêmes de l'amoureux ou l'amoureuse qui l'accompagnera pour cet évènement. Elle propose d'utiliser la méthode de Condorcet pour organiser le scrutin. La **méthode de Condorcet** est un système de vote qui permet de trouver le choix le plus consensuel, parmis une liste de propositions.

Maya étant informaticienne, elle a développé un logiciel qui lui affiche le vainqueur de Condorcet (ou plus exactement une version améliorée, comme nous le verrons pas la suite), en fonction des réponses qui ont été fournies par sa famille.

Saurez-vous **reproduire ce logiciel en langage C** ?

Quelques rappels :

La bibliothèque *string.h* vous permet de :

- comparer deux chaînes de caractères *s1* et *s2* avec la fonction *strcmp(s1, s2) qui* renvoie l'entier *0* lorsque les deux chaînes sont identiques ;
- récupérer la longueur d'une chaîne *s* avec la fonction *strlen(s),* qui renvoie la longueur de *s* ;
- et copier une chaîne *s2* dans une chaîne *s1* avec la fonction *strcpy(s1, s2).*

Exercice 1 : Structure de données

Chaque membre de la famille a eu pour consigne de rendre à Maya une version triée de la liste des personnes qu'elle fréquente, en l'ordonnant selon leur préférence. On représente ces listes en utilisant des listes chaînées, qui contiennent les prénoms (supposés uniques) des candidat·e·s.

Exemple : si un membre de la famille préfère Camille à Kaya, mais Kaya à Claude, la liste chaînée correspondante sera : [Camille] → [Kaya] → [Claude]

Q1) Définir la **structure *string_list_cell_s***, qui servira pour instancier les maillons de la liste chaînée. Utiliser ensuite cette structure et l'instruction *typedef*, pour définir le **type *list_t***, qui correspond à une liste de tels maillons et qui sera utilisé par la suite.

Q2) Implémenter la fonction qui retourne une nouvelle liste, qui contient le prénom *p :*

list_t list_cons(char p);*

Q3) Implémenter la fonction qui ajoute le prénom *p* à la liste *l*, sachant que ce nouveau prénom sera supposé avoir la préférence sur tous les autres déjà présents dans la liste (la *list_t* renvoyée correspond à la nouvelle tête de liste) :

list_t list_add(char p, list_t l);*

Q4) Implémenter la fonction permettant d'entièrement libérer la mémoire occupée par la liste *l* :

void list_free(list_t l);

Exercice 2 : Les duels de Condorcet

Le dépouillement du scrutin consiste à simuler l'ensemble des duels possibles, entre les prénoms. **Celui qui remporte tous ses duels a gagné.**

Exemple : dans le duel Camille versus Kaya, ce dernier est gagnant s'il est plus souvent classé devant Camille dans les listes, que l'inverse.

Q1) Implémenter la fonction qui retourne la valeur *1* lorsque le prénom *p1* remporte le duel face au prénom *p2* (ou qu'ils sont ex-æquo), en utilisant le tableau *t* qui contient les préférences des *N* membres de la famille (en cas de défaite de *p1*, la fonction retourne la valeur 0) :

int duel(char p1, char* p2, list_t t[N]);*

Vous êtes libre d'implémenter des fonctions auxiliaires, si vous le souhaitez.

Q2) Implémenter la fonction qui stocke dans *winner* le prénom correspondant au vainqueur de Condorcet, à partir du tableau *t* de préférences (il est possible qu'il n'y ait pas de vainqueur ou qu'il y en ait plusieurs : le prénom *"nobody"* sera alors utilisé) :

void condorcet(char winner, list_t t[N]);*

Exercice 3 : Paradoxe de Condorcet

Dans les cas où il n'y a pas de vainqueur, ou qu'il y a plusieurs vainqueurs ex-æquo, plusieurs solutions existent : il s'agit du paradoxe de Condorcet. Pour s'en sortir, **Maya a choisi d'utiliser la méthode Borda**. Lorsque cette méthode est utilisée à la suite de la méthode de Condorcet pour aider à désigner le vainqueur, **on parle alors de méthode Black, pour qualifier le duo**.

Borda propose d'attribuer un score à chacun des choix, et de désigner celui qui obtient le meilleur score, comme étant le vainqueur. Pour calculer le score du choix X, on compte le nombre de fois où X est arrivé en tête, que l'on multiplie par le nombre total de choix. Puis, on ajoute le nombre de fois où X est arrivé deuxième, multiplié par le nombre total de choix moins 1. Puis le nombre de fois où il est arrivé troisième, multiplié par le nombre total moins 2, etc.

Q1) Implémenter la fonction qui permet de calculer le score de Borda pour le prénom *p* :

int score(char p, list_t t[N]);*

Q2) Implémenter la fonction qui stocke dans *winner* le prénom correspondant au vainqueur de Borda (en cas d'égalité, utilisez de nouveau le prénom *"nobody")* :

void borda(char winner, list_t t[N]);*

Q3) Implémenter la fonction qui stocke dans *winner* le prénom correspondant au vainqueur de Black :

void black(char winner, list_t t[N]);*

Exercice 4 : Programme principal

Il ne reste plus qu'à écrire le programme principal en utilisant les fonctions implémentées. Il prendra cette forme (on ne vous demande pas de spécifier les *#include* nécessaires) :

```
#define N 4
int main() {
        /* Q1:  Instancier et initialiser quand nécessaire les variables utiles pour la suite, en respectant les
                noms proposés dans les énoncés précédents. */

        /* Q2:  Demander la saisie des listes ordonnées, qui ont été fournies par les membres de la famille
                (remplissage du tableau list_t t[N]). On ne demande pas de vérifier la cohérence des
                prénoms/listes saisies. */

        /* Q3:  Calculer et afficher le vainqueur de Black, correspondant au prénom de l'amoureux ou
                l'amoureuse qui accompagnera Maya à l'anniversaire de Cassy. */
}
```

On rappelle que la fonction à utiliser pour demander la saisie d'une chaîne de caractères *p* est la suivante :

scanf("%s", p);