Université Mohammed VI Polytechnique **Année 2021-2022**
Licence LSD, Semestre S_2 **Structures de données**

TRAVAUX DIRIGÉS
SÉRIE 1

1. Tableaux dynamiques

Exercice 1. *Notre but dans cet exercice est de créer un type, qu'on appelera* `tab`*, qui consiste en un tableau de réels dont la taille peut varier. Nous souhaitons aussi avoir des fonctions qui permettent de manipuler ce type de données (allocation-mémoire, saisie, affichage, concatenation,...). Pour cela nous définissons le type de données en question comme étant*

```
struct tab
{
    unsigned int taille;
    double * tete;
};
typedef struct tab tab;
```

Un objet T de type `tab` *est donc representé par sa taille n (le champ taille) ainsi qu'une zone mémoire pour stocker n réels de type* double *(le champ tete qui est un pointeur vers la zone mémoire).*

1. *Écrire la fonction* `creer()` *qui prend un entier positif n et doit renvoyer un* `tab` *T de taille n. Il ne faut bien sûr pas oublier d'écrire la fonction* `liberer()` *qui libère la mémoire allouée par la fonction* `creer()`*.*
2. *Écrire les fonctions* `saisie()` *et* `affichage()` *qui correspondent à ce type de données.*
3. *Écrire la fonction* `concatener()` *qui permet de concatener deux objets de type* `tab`*.*

2. Listes simplement chaînées

Exercice 2. *Soit L une liste simplement chainée de longueur n.*

1. *Écrire une fonction qui recheche un élément x dans la liste L. La fonction renvoie l'ardresse d'une cellule qui contient x s'il est présent dans L et NULL sinon.*
2. *Écrire une fonction qui prend en argument la liste L et un élément x et retourne le nombre d'occurences de x dans L.*
3. *Écrire une fonction qui prend L en argument et retourne le nombre d'éléments différents que la liste contient.*
4. *Évaluer la complexité de chaque fonction en termes de n.*

Date: February 27, 2022.

Exercice 3. *Soit L une liste simplement chainée de longueur n.*

1. *Écrire une fonction qui prend comme arguments L et un entier i positif, et supprime le i-ème élément de la liste. Si $i \geq n$ la fonction ne supprime aucun élément.*
2. *Écrire une fonction qui prend comme arguments L, un élément x et un entier positif i, et ajoute x à la i-ème position de la liste. Si $i \geq n$ la fonction ajoute x à la fin de la liste L.*

Exercice 4. *(Conversions entre tableaux et listes) On étudie dans cet exercice comment convertir un tableau en liste simplement ou doublement chaînée.*

1. *Écrire une fonction* `tableauListe()` *qui prend en entrée un tableau de taille n et le transforme en liste simplement chaînée.*
2. *Écrire une fonction* `listeTableau` *qui prend en entrée une liste simplement chaînée et la transforme en un tableau.*
3. *Étudier la complexité de chacune des deux conversions.*

Exercice 5 (Concaténation de deux listes)**.** *Étant données deux listes L_1 et L_2, la concaténation de L_1 et L_2 consiste à ajouter L_2 à la fin de L_1. Ainsi, si $L_1 = [1, 2, 3]$ et $L_2 = [4, 5, 6]$ leur concaténation est la liste $[1, 2, 3, 4, 5, 6]$.*

1. *Écrire une fonction qui permet de concaténer deux listes simplement chaînées.*
2. *Écrire une fonction qui permet de concaténer deux listes doublement chaînées.*

Exercice 6. *On se propose de réaliser le tri par insertion sur une liste simplement chaînée d'entiers.*

1. *Écrire une fonction* `inserer()` *qui prend en entrée une liste chaînée L triée par ordre croissant et un entier x, et retourne la liste L dans laquelle x est inseré de manière à ce qu'elle reste triée.*
2. *En utilisant la fonction* `inserer()`*, écrire une fonction* `trier()` *qui prend en entrée une liste chaînée L et retourne une liste chaînée L_1 qui contient les mêmes éléments que L, mais elle est triée par ordre croissant.*

3. Piles

Exercice 7. *Un problème fréquent dans les éditeurs de texte consiste à vérifier si les parenthèses dans une chaîne de caractères sont balancées et correctement incluses l'une dans l'autre. Par exemple, la chaîne (())()((())) est correctement formatée mais (())(() ne l'est pas.*

On cherche à écrire une fonction qui prend en entrée une chaîne de caractères et décide si les parenthèses dans la chaîne sont correctement formatées. Pour ceci, on procède de la manière suivante.

1. *On déclare une pile P vide au départ. La nature des éléments de la pile ne compte pas, et on peut supposer qu'il s'agit d'entiers.*
2. *À chaque lecture d'un caractère de la chaîne trois possibilités se présentent.*
   - *Le caractère lu n'est pas une parenthèse. Dans ce cas on ne fait rien.*
   - *Le caractère lu est la parenthèse ouvrante "(". Dans ce cas on empile 1 dans la pile P.*

- *Le caractère lu est une parenthèse fermante ")". Dans ce cas, on a les cas suivants.*
  - *Si la pile est vide alors on vient de trouver une parenthèse fermante qui ne correspond à aucune parenthèse ouvrante. La chaîne n'est donc pas bien formatée.*
  - *Si la pile est non vide alors la prenthèse fermante lue correspond à la dernière parenthèse ouvrante lue. Dans ce cas on dépile P.*

Écrire une fonction qui prend en entrée une chaîne de caractères et utilise la méthode décrite ci-dessus pour décider si les parenthèses dans la chaîne sont bien formatées.

Exercice 8 (Copie d'une pile et d'une file)**.** *On souhaite programmer des fonctions qui permettent de copier une pile donnée P, tout en la conservant.*

1. *Écrire une fonction* `copie_inverse()` *qui prend en entrée une pile et renvoie une copie de cette pile dans laquelle l'ordre des élément est inversé.*
2. *Écrire une fonction* `copie()` *qui prend en entrée une pile et renvoie une copie de cette pile.*

4. Files

Exercice 9. *Écrire une fonction qui prend en entrée une file et permet d'en faire une copie. La file fournie à la fonction doit bien entendu être conservée.*

Exercice 10. *Considérons une file d'attente devant un cinéma. La file initialement vide se remplit au fur et à mesure que les individus arrivent. On suppose que si un nouvel individu aperçoit dans la file un ami, alors il se joint à lui pour attendre.*

Pour manipuler cette file d'attente, on considérera que les individus sont représentés par des entiers. Deux amis seront alors deux entiers identiques. La structure de donnéees utilisée pour représenter la file devra donc intégrer non seulement l'individu, mais aussi le nombre d'occurrences associé.

1. *Proposer une structure de données qui permet de gérer cette file d'attente.*
2. *Écrire une fonction qui permet d'ajouter un nouvel individu dans la file d'attente.*
3. *Écrire une fonction qui permet de retirer le premier individu dans la file d'attente.*

5. Implémentation d'une pile et d'une file par un tableau

Exercice 11. *En utilisant la structure de données tableau implémenter la structure de données pile. L'implémentation doit fournir au moins les quatre opérations suivantes :*

`pileVide`	:	*Permet de créer une pile vide.*
`estVide()`	:	*Teste si une pile est vide.*
`lire()`	:	*Permet de lire l'élément au sommet de la pile.*
`empiler()`	:	*Permet d'empiler un élément dans une pile.*
`depiler()`	:	*Permet de dépiler une pile.*

Évaluer la complexité temporelle de chacune des opérations.

Exercice 12. *En utilisant la structure de données tableau implémenter la structure de données file. Comme pour le cas de la pile, l'implémentation doit fournir au moins les quatre opérations suivantes :*

`fileVide` : *Permet de créer une file vide.*

`estVide()` : *Teste si une file est vide.*

`lire()` : *Permet de lire l'élément au début d'une file.*

`enfiler()` : *Permet d'enfiler un élément dans une file.*

`defiler()` : *Permet de défiler une file.*

Évaluer la complexité temporelle de chacune des opérations.

Exercice 13. *On se propose de modéliser la gestion des patients dans un cabinet médical. Un* `patient` *est caractérisé par son* `nom`*, son* `prénom` *et un champ* `rdv` *de type entier qui vaut* 0 *si le patient n'a pas de rendez-vous et* 1 *si le patient a un rendez-vous.*

Chaque patient est placé, à son arrivée, dans une salle d'attente qui est modélisée par une liste simplement chaînée de patients. Les patients ayant un rendez-vous voient le medecin selon leur ordre d'arrivée. Une fois il n'y a plus de patient ayant un rendez-vous le medecin reçoit les patients n'ayant pas de rendez-vous selon leur ordre d'arrivée.

1. *Écrire la structure de donnée* `patient`*, et ensuite la structure* `cellule` *qui permet de construire une liste chaînée de patients.*
2. *Écrire la fonction* `ajoutPatient()` *qui permet d'ajouter un nouveau patient P à une liste chaînée L de patients.*
3. *Écrire la fonction* `extrairePatient()` *qui permet d'extraire le patient dont le tour est venu pour consulter le medecin. Ainsi, si L est la liste des patients, la fonction* `extrairePatient()` *supprime de L le premier patient et le retourne s'il n'y a aucun patient dans la liste ayant un rendez-vous. Sinon, la fonction supprime de L le premier patient ayant un rendez-vous et le retourne.*