Université de Saïda Dr. Moulay Tahar
Département de mathématiques
1ère L.M.D–MI

Correction de l'Examen Final

Dimanche 07 Juillet 2019 de 08h00 à 10h00

Aucun document n'est autorisé
Les solutions doivent être rédigées en langage algorithmique ou en C.
Tout appareil électronique doit être éteint (Téléphone, Ordinateur, Tablette, etc.).

Exercice 1 (04,00 points, 20')

Question 1 : Quelle est la différence entre fonction et procédure ? **(1,00 pt)**

Question 2 : L'opération d'ajout d'une cellule en queue de liste est un peu plus compliquée que l'insertion en tête de liste (quoi ?) **(1,00 pt)**

Question 3 : Qu'est-ce qu'une liste doublement chaînée ? **(1,00 pt)**

Question 4 : Les piles et les files sont deux structures de données utilisées en informatique. Quelle est la différence entre une Pile et une File ? **(1,00 pt)**

Réponses

Réponse 1 : Une fonction est un sous-algorithme qui, à partir de donnée(s), calcul et rend à l'algorithme un et un seul résultat (une seule valeur) alors qu'en général, une procédure affiche le(s) résultat(s) demandé(s). **(1,00 pt)**

Réponse 2 : Elle nécessite un parcours de la liste pour chercher l'adresse du dernier élément (queue de la liste). **(1,00 pt)**

Réponse 3 : Une liste doublement chaînée est une liste chaînée, dans laquelle on peut accéder à l'élément prédécesseur. **(1,00 pt)**

Réponse 4 : Dans une pile, l'élément supprimé est le dernier inséré (*LIFO* ou *Last In First Out*) et dans une file, l'élément supprimé est le plus ancien (*FIFI*, *First In Fisrt Out*) **(1,00 pt)**

Exercice 2 (10,00 points, 60')

Un fichier contient des descriptions de code article et de prix. Le code d'article est un numéro entre 0 et 99. Chaque ligne du fichier contient un code et un prix séparés par un espace :

```
code1 prix1
code2 prix2
etc.
```

On se propose d'organiser les données en mémoire centrale sous forme de tableaux : on trouve le prix de chaque produit de code `c` dans la case de d'indice `c` du tableau.

a. Proposer une structure de données pour représenter les produits en mémoire centrale. **(1,00 pt)**

b. Écrire une fonction qui réalise le chargement en mémoire centrale des données. **(2,00 pt)**

c. Écrire une fonction qui donne le prix d'un produit à partir de son code. **(1,00 pt)**

d. Écrire une fonction qui permet à un utilisateur de rentrer un nouveau produit dans la base. On supposera que toute les données (la base) ont été préalablement chargées en mémoire. **(2,00 pt)**

e. Écrire une fonction qui permet de sauvegarder les données (la base) dans un fichier. **(2,00 pt)**

f. Écrire l'algorithme principal qui charge les données et, tant que l'utilisateur le souhaite, propose un menu avec les trois options : Ajouter un produit, Consulter un prix, Quitter. En din d'algorithme, la sauvegarde des données sera effectuée. **(2,00 pt)**

Solution

---

a. (1,00 pt)

```
Structure Produit
{
    prix Tableau[100] : reels
    valide Tableau[100] : entiers        // valide[.]=1 si le produit existe
}
```

b. (2,00 pt)

```
Fonction chargement() : Produit
1  Var i, code : entiers
2      prix : reel
3      base : Produit
4  Debut
5      Ouvrir ("produit.txt", 1, Lecture)
6      Pour i allant de 0 a 99 Faire
7          base.valide[i] ← 0
8      FinPour
9      Repeter
10         LireFichier(1, code, prix)
11         base.valide[code] ← 1
12         base.prix[code] ← prix
13     Jusqu'a (EOF(1))
14     Fermer(1)
15     Retourner(base)
16 Fin
```

c. (1,00 pt)

```
Fonction prix (base : Produit, code : entier) : reel
1 Debut
2 Si (base.valide[code]= 1) Alors
3    Retourner (base.prix[code])
4 Sinon
5    Retourner(−1.0) // prix inconnu
6 FinSi
7 Fin
```

d. (2,00 pt)

```
Fonction ajout(base : Produit) : Produit
1 Var code : entier
2       prix : reel
3 Debut
4       Ecrire("Entrer un code et un prix")
5       Lire(code, prix)
6       base.valide[code]← 1
7       base.prix[code]← prix
8       Retourner(base)
9 Fin
```

e. (2,00 pt)

```
Procedure sauve(base : Produit)
1 Var i : entier
2 Debut
3       Ouvrir ("produit.txt", 1, Ecriture)
4       Pour i allant de 0 a 99 Faire
5           Si (base.valide[i]= 1) Alors
6             EcrireFichier(1, i, base.prix[i])
7           FinSi
8       FinPour
9       Fermer(1)
10 Fin
```

f. (2,00 pt)

```
Algorithme Menu
1 Var code : entier
2       c : caractere
3       base : Produit
4 Debut
5       base ← chargement()
6       TantQue(c<>'3') Faire
7               Ecrire("Voulez-vous")
8               Ecrire("1- Ajouter un produit, 2- Consulter un prix, ou 3- Quitter")
11              c ← '0'
12              TantQue((c<>'1') et (c<>'2') et (c<>'3')) Faire
13                      Ecrire("Votre choix ?")
14                      Lire(c)
15              FinTantQue
16              Selon(c)
17                 Cas '1' :
18                         base ← ajout(base)
19                 Cas '2' :
20                         Ecrire("code ?")
21                         Lire(code)
22                         Ecrire("Le prix est de : ", prix(base, code))
23                 Sinon // Quitter
24               FinSelon
25      FinTantQue
26 Fin
```

Exercice 3 (06,00 points, 40')

a. Écrire une fonction qui prend en paramètre une liste chaînée d'entiers et vérifie qu'elle est triée dans l'ordre croissant. **(3,00 pt)**

b. Écrire une fonction qui insère un élément dans une liste chaînée triée. La liste retournée doit être triée. Pour cela, on recherchera l'adresse de la cellule (si elle existe) juste avant l'emplacement de l'insertion. **(3,00 pt)**

Solution

---

a. (3,00 pt)

```
// La fonction retourne 0 si la liste est triée, −1 si elle est vide et 1 si elle n'est pas triée
Fonction VerifieTrie(Var *L : Cellule) : entier
1  Var *p, *suivant : Cellule
2  Debut
3      Si (L=NULL) Alors
4        Ecrire("La liste est vide")
5        Retourner(−1)
6      FinSi
7      p← L
8      suivant← p->suivant
9      Si (suivant=NULL) Alors
10       Retourner(0)
11     FinSi
12     TantQue((suivant->suivant<>NULL) et (p->donnee < suivant->donnee)) Faire
13           p ← p->suivant
14           suivant ← p->suivant
15     FinTantQue
16     Si (suivant->suivant=NULL) Alors
17       Si (p->donnee < suivant->donnee) Alors
18         Retourner(0) // La liste est triée
19       Sinon
20         Retourner(1)
21       FinSi
22     FinSi
23     Retourner(1)
24 Fin
```

b. (3,00 pt)

```
Fonction InsereElement* (Var *L : Cellule, donnee : TypeDonnee) : Cellule
1  Var *p, *nouveau, *suivant : Cellule
2  Debut
3       Allouer (nouveau)
4       nouveau->donne ← donne
5       nouveau->suivant ← NULL
6       Si (L=NULL) Alors
7         Retourner(nouveau)
8       FinSi
9       p← L
10      suivant← p->suivant
11      Si (p->donnee < donnee) Alors // Si la première donnée est < que la donnée à insérer
17        Si (suivant= NULL) Alors
18          p->suivant ← nouveau
18          Retourner (L)
17        FinSi
8       Sinon // Si la première donnée est > à la donnée à insérer
18          nouveau->suivant ← p
18          Retourner (nouveau)
8       FinSi
12      TantQue((suivant->suivant<>NULL) et (suivant->donnee < donnee)) Faire
13              p ← p->suivant
14              suivant ← p->suivant
15      FinTantQue
16      Si (suivant->donnee > donnee) Alors
18          p->suivant ← nouveau
18          nouveau->suivant ← suivant
16      Sinon
18          suivant->suivant ← nouveau
16      FinSi
22      Retourner(L)
2  Fin
```