UNIVERSITE DE TUNIS EL MANAR
ECOLE NATIONALE D'INGENIEURS DE TUNIS

EXAMEN Programmation Procédurale

Session : Principale	**Classes :** 1AGE1, 1GHE, 1Minds
Enseignants : L. Haj Massaoud C. Jelassi	**Date :** 11/01/2018
Durée: 2H	**Documents:** Non autorisés

Remarque : Les questions sont indépendantes dans une large mesure. Vous pouvez utiliser les fonctions écrites dans les questions précédentes et les fonctions qu'on suppose avoir sans les réécrire.

Problème : Manipulation des listes simplement chainées pour un jeu de cartes

On distribue à égalité les cartes d'un jeu aux 2 joueurs qui les rassemblent en paquet, face cachée, devant eux. On appellera une **main** le paquet d'un joueur. Les valeurs d'une carte varient de **1 à 9**.

Chaque joueur tire la première carte de sa main et la pose sur la table, face découverte ; On appellera les cartes posées sur la table les cartes de bataille. Celui qui a la carte la plus forte ramasse les 2 cartes et son score s'incrémente de 1.

Pour simuler le jeu, les structures de données utilisées sont des listes simplement chainées.

Les cartes de main de chaque joueur peuvent être représentées par une **liste simplement chainée**. Le joueur ne peut prendre que la première carte de sa main et s'il ajoute, il ajoute uniquement à la fin de la liste.

Les cartes de bataille sont aussi une **liste simplement chainée**, le joueur regarde et compare toujours la première carte mais il ajoute seulement en début de la liste.

Quand un joueur gagne, il reprend les cartes une par une à partir du début de la liste de bataille et il les ajoute à la fin sa liste de main.

Dans le cadre de ce problème, on se limitera seulement à la manipulation des deux listes simplement chainées ; liste de **main** et liste de **bataille** sans entrer dans les détails ; on ne s'intéresse pas à la distribution de cartes et aux règles du jeu.

Question 1 : Définition et gestion d'une carte (4 points)

Une carte est donc définie dans notre problème par une valeur (**int**), une couleur (**char :** '**R**' si elle est rouge ou '**N**' si elle est noire) et sa visibilité (**char** : '**c**' si elle est cachée ou '**v**' si elle est visible)

On suppose avoir la fonction permettant de remplir les champs correspondant à une carte. Son prototype est :

```
void SaisieC(Carte* c)
```

1.1 Définir la structure de donnée **Carte** permettant de représenter et manipuler les cartes.

1.2 Ecrire une fonction permettant d'afficher une carte en mode texte. Son prototype est :

```
void AfficheC(Carte c)
```

1.3 Ecrire une fonction permettant de comparer 2 cartes. Cette fonction retourne un nombre négatif (-1) si la valeur de la première carte est plus petite que la valeur de la deuxième, 0 si elles sont égales, un nombre positif (1) si la valeur de première est plus grande que celle de la deuxième. Son prototype est :

```
int Compare(Carte c1, Carte c2)
```

Question 2 : Gestion des batailles (6 points)

Soit la structure suivante :

```
typedef struct
{
  Carte cte;
  struct ListeBataille *suivant;
}ListeBataille;
```

2.1. Créer une liste vide :

```
ListeBataille* Creer_ListeB()
```

2.2. Tester si une liste de bataille est vide, la fonction retourne 0 si la liste est vide et 1 sinon

```
int B_vide(ListeBataille* lb)
```

2.3. Ajouter une carte en début de la liste de bataille:

```
ListeBataille* AjouterCarteLB(Carte c, ListeBataille *lb)
```

2.4. Renvoyer la première carte de la liste de bataille. La fonction renvoie la première carte sans la supprimer de la liste:

```
Carte PremiereCarteB(ListeBataille* lb)
```

2.5. Supprimer la première carte de la liste. La fonction renvoie la liste après suppression :

```
ListeBataille * SupprimerCarteB(ListeBataille * lb)
```

Question 3 : Gestion des mains (4 points)

Soit la structure suivante :

```
typedef struct
{
  Carte cte;
  struct ListeMain *suivant;
} ListeMain;
```

Dans la main d'un joueur, on ajoute des cartes à la fin et on retire des cartes en début de la liste. On suppose avoir les fonctions suivantes :

`ListeMain* Creer_Main()` qui crée une liste vide pour une main,
`int M_vide(ListeMain* lm)` qui teste si une main est vide,
`Carte SupprimerCarteM(ListeMain* lm)` qui supprime et renvoie la première carte d'une main.

3.1. Ecrire la fonction qui ajoute une carte en fin de liste :

```
ListeMain* AjouterCarteM(Carte c, ListeMain* lm)
```

3.2. Ecrire la fonction qui affiche toutes les cartes d'une main :

```
void visualiser_Main(ListeMain* lm)
```

Question 4 : Définition et gestion d'un joueur (4 points)

Un joueur est défini dans notre problème par son **pseudo** (char*), sa **main** (ListeMain*) et son **score** (int)

4.1. Définir la structure de donnée **Joueur** permettant de représenter et manipuler les joueurs.

4.2. Ecrire une fonction qui saisit les informations relatives à un joueur. On initialisera impérativement sa main à NULL et son score à 0. Son prototype est :

```
void SaisieJ(Joueur* j)
```

4.3. Ecrire une fonction permettant d'afficher le pseudo et la main d'un joueur donné en mode texte. Son prototype est :

```
void AfficheJ(Joueur j)
```

Question 5 : Sauvegarde des mains (2 points)

5.1. Ecrire une fonction qui permet de sauvegarder la main d'un joueur donné dans un fichier. Son prototype est :

```
void SaugegarderMain (Joueur aj, FILE * pf)
```