Nom et Prénom : _____

DM II

Hicham Janati

À rendre au plus tard le 18 Avril
en un seul fichier (.txt, .doc, .pdf)

Traitez plusieurs parties pour que je puisse évaluer vos connaissances dans tous les chapitres.

1 Exercices

Pour chaque fonction, le prototype est donné en guise d'indication. Vous devez néanmoins le respecter. Certaines questions sont en chaîne.

1.1 Logique, Tableaux, Pointeurs

1. **Écrire une fonction qui renvoie le dernier chiffre (chiffre des unités) d'un entier a. Unite(456) renvoie 6. Unite(3) renvoie 3.**

```
1 int Unite(int a);
```

2. **Écrire une fonction qui renvoie le nombre de chiffres d'un entier a. Longueur(8560) renvoie 4.**

```
1 int Longueur(int a);
```

3. **Écrire une fonction qui renvoie la puissance p-ème d'un entier a.**

```
1 int Puissance(int x, int  p);
```

4. **Écrire une fonction qui prend un entier a et un entier i en arguments et renvoie le i-ème chiffre du nombre a en partant de droite à gauche : i = 0 correspond aux unités, i = 1 aux dizaines etc. Chiffre(435,0) renvoie 5. Chiffre(435,1) renvoie 3. Chiffre(435,2) renvoie 4. Chiffre(435,i) renvoie 0 pour $i \geq 3$.**

```
1 int Chiffre(int a, int i);
```

5. **Écrire une fonction qui prend un entier et un tableau (et sa taille) en arguments et le remplit par les chiffres de a. Si a = 4593, le tableau doit être de la forme [0—0—....—0—4—5—9—3]. Les unités à la dernière case, les dizaines à l'avant dernière etc.**

```
1 void Decouper(int a, int T[], int taille);
```

6. **Même question avec un pointeur sur T (il est tout à fait possible de considérer T comme un tableau ou un pointeur dans les deux cas, dans cette question j'entends une manipulation du tableau avec le formalisme pointeurs) :**

```
1 void Decouper2(int a, int *T, int taille);
```

7. **Écrire une fonction qui fait l'opération inverse : elle prend un tableau T d'entiers et renvoie le nombre formé. Comme à la question précédente, le tableau contient le nombre de droite à gauche.**

```
1 int Rassembler(int *T, int taille);
```

8. **Écrire une fonction qui prend deux entiers a et i et renvoie l'entier a en mettant à 0 son i-ème chiffre. Azero(4560,1) renvoie 4500. Azero(23493,4) renvoie 3493. Azero(984,0) renvoie 980. Azero(4051,i) renvoie 4051 pour $i \geq 4$.**

```
1 int Azero(int a, int i);
```

1.2 Chaînes, pointeurs

Sans utiliser string.h !

9. **Écrire une fonction qui prend une chaîne de caractères M et entier i en arguments et tronque M après la i-ème lettre.**

```
1 void Troncature(char M[], int i);
```

10. **Écrire une fonction qui renvoie la taille d'une chaîne M**

```
1 int LongueurChaine(char *M);
```

11. **Écrire une fonction qui compare deux chaînes de caractères et renvoie un pointeur sur la plus longue d'entre elles.**

12. **Écrire une fonction qui compare deux chaînes de caractères et renvoie un pointeur sur la plus longue d'entre elles.**

```
1 char* PlusLongue(char *U, char*V );
```

13. **Écrire une fonction qui remplit une chaîne Z formée par la concaténation de deux chaînes U et V où la plus petite est à la fin**

```
1 void CopierChaine(char *U, char*V, char* Z);
```

2 Problèmes

2.1 Un jeu de cartes

Imaginons un jeu de cartes avec 20 numéros [1-20] et deux couleurs [rouge - noir]. En bataille, Une carte avec le numéro plus grand l'emporte toujours. En cas d'égalité, le rouge domine. En cas d'égalité du numéro et de la couleur, on rejoue le tour en question. Vous jouez contre l'ordinateur. Et à chaque itération :

— Chacun de vous pioche une carte du lot.

— Après avoir observé votre carte (uniquement la votre), vous devez saisir un montant à miser.

— Après l'avoir saisi, la carte de l'adversaire est révélée. Elle est comparée à la votre pour déterminer le vainqueur du face-à-face.

— Votre gain est calculé de la façon suivante : Si votre mise est égale à m, vous gagnez (ou perdez) (20 - i)*m où i est le numéro de la carte (Si vous gagnez avec i = 1, vous gagnez 19*m, logique non ? c'est très improbable de gagner avec la carte 1)

1. **Vous êtes désormais en mesure de proposer votre propre solution en définissant les fonctions que vous jugez pertinentes. Faîtes attention à la fonction qui génère les cartes piochées par exemple : une même carte ne peut pas être piochée deux fois ! Le jeu s'arrête lorsque l'on a joué toutes les cartes : donc après 20 tours (à chaque tour, deux cartes sont piochées).**

2.2 Structures et liste chaînées

Toujours dans le même monde des cartes, nous allons définir une structure *Carte* comportant deux attributs : un entier *numero* et une chaîne *couleur*. Je vous recommande de revoir le cours `https://hichamjanati.github.io/media/C/structures.pdf` pour vous rafraîchir la mémoire.

1. **Créez un fichier.h où vous allez définir la structure *Carte* précédée de l'alias avec *typedef***

2. **Écrire une fonction qui prend un pointeur sur une Carte C, un entier n, et une chaîne s et affecte à C les attributs n et s.**

```
1 void Affectation(Carte *C, int n, char *s);
```

3. **Écrire une fonction qui prend deux structures en paramètres et les compare. Elle renvoie 1 si la première est plus *forte*, 2 si la deuxième est plus forte, et 0 si elles sont identiques (Voir l'énoncé du problème précédent pour comprendre comment effectuer la comparaison**

```
1 int Compare(Carte A, Carte B);
```

4. **Nous allons maintenant simuler une vraie *main* avec une liste chaînée. Modifiez la structure Carte en ajoutant un troisième attribut : un pointeur sur une Carte.**

5. **Le principe est simple, nous voulons que la liste comporte 3 éléments toujours dans l'ordre : la plus petite carte pointe sur la carte moyenne qui pointe sur la plus grande qui pointe sur NULL. Créez une fonction qui prend un pointeur sur la liste L (qui est un pointer sur la première carte), un pointeur sur une carte C et insère la carte C à la bonne place.**

```
1 void Inserer(Carte *L, Carte *C);
```

6. **Créez une fonction qui prend un pointeur sur la liste et supprime l'élément le plus petit (donc le premier). Elle doit renvoyer un pointeur sur le nouvel élément *tête* pour ne pas perdre l'adresse de la liste !**

```
1 Carte* Supprimer(Carte* L);
```