DM II - Corrigé

Hicham Janati

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.**

```
int Unite(int a){
  return a%10;
}
```

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

```
int Longueur(int a);
  int l=0;
  while(a !=0){
    a/=10;
    l++,
  }
  return l;
 }
```

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

```
int Puissance(int x, int  p){
  if(p==0) return 1;
  return x*Puissance(x,p-1);
}
```

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 $i \geq 3$.**

```
  int Chiffre(int a, int i){
  int n = Longueur(a);
  // Si i dépasse la taille du nombre, on renvoie 0
  if(i>= n) return 0;
```

```
5
6    // Sinon, on divise par 10^i pour ramener le ième chiffre à la position 0
7    a /= Puissance(10,i);
8
9    // On prend les unités de ce nouveau nombre
10
11   return Unite(a);
12 }
```

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){
2   int i;
3   for(i=N-1;i>=0;i--)
4   T[i] = Chiffre(a,N-i);
5 }
```

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){
2   int i;
3   for(i=N-1;i>=0;i--)
4   *(T+i) = Chiffre(a,N-i);
5 }
```

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{
2   int i,a=0;
3   for(i=0;i<N;i++){
4   a+= Puissance(10,i)*T[i];
5   }
6   return a;
7 }
```

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){
2 int n = Longueur(a);
3 int T[n];
4 Decouper(a,T,n);
5 T[N-i] = 0;
6 return Rassembler(T,n);
7 }
```

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.**

```c
void Troncature(char M[], int i){
  M[i] = '\0';
}
```

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

```c
int LongueurChaine(char *M){
  int i=0;
  while(M[i]!='\0') i++;
  return i;
}
```

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

```c
char* PlusLongue(char *U, char*V ){
  if(LongueurChaine(U)>=LongueurChaine(V)) return U;
  return V;
}
```

12. **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**

```c
void CopierChaine(char *U, char*V, char* Z){
  int i=0;
  // la plus longue est:
  char* L = PlusLongue(U,V);
  // la plus petite est:
  char* P = U+V-L;

  // on se positionne à la fin de L
  while(L[i]!='\0'){
  Z[i] = L[i];
  i++;
  }
  int j = 0;
  While(P[j]!='\0'){
  Z[j+i] = P[j];
  j++;
  }
  Z[j+i] = '\0';
}
```

2 Problèmes

2.1 Un jeu de cartes

2.1.1 Enoncé

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).**

2.1.2 UNE solution ...

Nous avons 40 cartes que l'on peut ordonner comme suit :
1 noire < 1 rouge < 2 noire < 2 rouge ... < 20 noire < 20 rouge.
Ainsi, on peut très bien représenter cet ordre avec des floats en utilisant l'équivalence :

$$1.0 \leftrightarrow 1 \text{ noire}$$
$$1.5 \leftrightarrow 1 \text{ rouge}$$
$$\ldots\ldots$$
$$13.0 \leftrightarrow 13 \text{ noire}$$
$$13.5 \leftrightarrow 13 \text{ rouge}$$

Comme la partie compte un nombre de tours fixé, stockons nos cartes dans un tableau :

$$|1.0|1.5|2.0|2.5....|19.5|20.0|20.5|$$

Maintenant, pour que notre générateur aléatoire soit sans remise (c à dire que toute carte ne peut être piochée qu'une seule fois par partie) nous pouvons tout simplement "mélanger" les cartes au début puis les tirer une à une ! Ainsi, nous allons permuter notre tableau puis lire les cartes dans l'ordre du tableau.
J'ai ajouté une fonction AfficherTableau que vous pouvez insérer là où vous voulez dans le main afin de visualiser ce qui se passe.

Fichier main.c

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include"fonctions.h"

int main(){
  int jouer = 1, N = 40;

  srand(time(NULL));
  printf("------ Bienvenue au jeu de cartes ! ------- \n Voulez-vous jouer (1/0) ?\n")
```

```
11    scanf("%d",&jouer);
12
13    while(jouer){
14      int tour,mise, resultat, solde=0;
15      float carte, carteIA;
16
17      // On crée un tableau de 40 cartes
18      float TabCartes[N];
19
20      // On remplit le tableau
21      Initialisation(TabCartes,N);
22
23      // On permute les valeurs du tableau aléatoirement
24      Permute(TabCartes,N);
25
26      for(tour=0;tour<N/2;tour++){
27
28      printf("------ Tour numero %d ------ \n", tour+1);
29
30      // On pioche votre carte
31      carte = Piocher(TabCartes,N,2*tour);
32
33      // On l'affiche
34      AfficherCarte(carte);
35
36      printf("Saisissez une mise m:\n");
37      scanf("%d",&mise);
38
39      // On pioche la carte de l'IA et on l'affiche
40      carteIA = Piocher(TabCartes,N,2*tour+1);
41      AfficherCarte(carteIA);
42
43      int numero = (int)(carte);
44      if (carte > carteIA){
45        printf("Face à face gagné ! Vous remportez %d", mise*(20-numero));
46        solde += mise*(20-numero);
47        }
48      else {
49        printf("Face à face perdu :( Vous perdez %d", mise*(20-numero));
50        solde -= mise*(20-numero);
51        }
52      }
53
54      printf("------------------- C'est fini ! -------------- \n");
55      if (solde>0) printf("Vous avez gagné %d", solde);
56      else printf("Vous avez perdu %d", solde);
57
58      printf("----- Une autre partie (1/0) ?\n");
59      scanf("%d", &jouer);
60
61    }
62  }
```

Fichier fonctions.h

```
1
2  void Initialisation(float TabCartes[], int N);
```

```
3 void Permute(float TabCartes[], int N);
4 void AfficherTableau(float TabCartes[], int N);
5 float Piocher(float TabCartes[], int N, int tour);
6 void AfficherCarte(float carte);
```

Fichier fonctions.c

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<time.h>
4 void Initialisation(float TabCartes[], int N){
5   int i;
6   // On remplit le tableau par les valeurs 1 1.5 2 2.5 ... 20.5
7   for(i=0;i<N;i++) TabCartes[i] = (float)(i+2)/2;
8 }
9 void Permute(float TabCartes[], int N){
10  int i,j;
11  float aux;
12  // On permute le tableau
13  for(i=0;i<N;i++){
14    j = rand()%N;
15    aux = TabCartes[j];
16    TabCartes[j] = TabCartes[i];
17    TabCartes[i] = aux;
18    }
19 }
20
21
22 void AfficherTableau(float TabCartes[], int N){
23   int i;
24   printf("\n---- Le tableau -------:\n");
25   for(i=0;i<N;i++) printf("%f\n",TabCartes[i]);
26   printf("\n------------------------\n");
27   }
28
29 float Piocher(float TabCartes[], int N, int tour){
30   float carte;
31   // on génère un indice aléatoire entre 0 et N-tour (numéros piochables)
32   carte = TabCartes[tour];
33   // on bascule le numéro pioché à la fin du tableau
34   TabCartes[tour] = TabCartes[N-tour-1];
35   TabCartes[N-tour-1] = carte;
36
37   return carte;
38 }
39
40 void AfficherCarte(float carte){
41   int c = 2*carte;
42   if (c%2==0) printf("La Carte piochée: %d - noire \n", c/2);
43   else    printf("La Carte piochée: %d - rouge \n", c/2);
44
45 }
```

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***

```
typedef Struct Carte Carte:
struct Carte{
  int numero;
  char *couleur;
}
```

2.

3. **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.**

```
void Affectation(Carte *C, int n, char *s){
  C->numero = n;
  int i = 0;
  while(s[i]!='\0'){
  (C->couleur)[i] = s[i];
  i++;
  }
}
```

4. **Erreur d'énoncé : le cas de cartes identiques est impossible car les cartes sont uniques 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)**

```
int Compare(Carte A, Carte B){
  if(A.numero > B.numero)
    return 1;

  if(A.numero == B.numero){
    if(A.couleur[0] == 'r') return 1;
    }
  return 2;
}
```

5. **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.**

```
struct Carte{
  int numero;
  char* couleur;
  Carte* suivante;
}
```

6. **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){
2  /* Pour insérer la carte à la bonne place nous devons parcourir la liste
3  jusqu'à trouver un élément suivant supérieur à C. Ainsi, pour que nous
4  puissions comparer l'élément L, il nous faut une carte dite tête de liste:
5  Il s'agit d'une carte vide qui ne sert qu'à contenir un pointeur sur la
6  première structure de notre liste.
7  */
8
9  Carte* Tete;
10 Tete->suivante = L;
11
12 // On crée un curseur initialisé à Tête:
13
14 Carte* Curseur = Tete;
15
16 /* Tant que la liste
17 n'est pas finie: */
18 while(Curseur->suivante != NULL){
19   if (Comparer(Curseur->suivante,C)==1){
20   // On insère C:
21     C->suivante = Curseur->suivante;
22     Curseur->suivante = C;
23     break;
24   }
25   // On avance:
26   Curseur = Curseur->suivante
27 }
28 if (Curseur->suivante == NULL){
29 C->suivante = NULL;
30 Curseur->suivante = C;
31 }
32 }
```

7. **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){
2  /* Il suffit d'"oublier" le premier élément, et de renvoyer un pointeur sur la deuxièm
3
4  return L->suivante;
5  }
```