ENSEIRB-MATMECA – Filière Electronique

Examen Langage C - PG108
2019-2020 1ère session
Mardi 7 Janvier 2020
Durée de l'épreuve : 2h / documents autorisés – sans calculatrice

1) Fonctions mystère:

Que font les fonctions suivantes ?
Pour chaque fonction, maximum : une phrase pour expliquer ce que fait la fonction, une phrase pour l'utilité des paramètres.

```
int joe(char *rantanplan) {
     if (rantanplan[0] < 'A')
          return 0;
     if (rantanplan[0] > 'Z')
          return 0;
     return 1;
}
```

```
int jack(int *jolly, int jumper) {
     for (int ma=0; ma<jumper – 1; ma++)
          if (jolly[ma] > jolly[ma + 1])
               return 0;
     return 1;
}
```

```
double william(double *lucky, int luke) {
     float rantanplan = 0.0;
     int ma = luke;
     while (ma -- )
          rantanplan += lucky[ma]*lucky[ma];
     return sqrt(rantanplan);
}
```

```
void averell(int *billy, int *kid, int calamity) {
     if (calamity) {
          if (*billy < *kid) {
               int jane = *billy;
               *billy = *kid;
               *kid = jane;
          }
          averell(billy + 1, kid + 1, calamity – 1);
     }
}
```

2) Exercice:

Écrivez un programme complet qui récupère deux entiers comme arguments en ligne de commande, et qui affiche leur somme dans le terminal.

3) Problème:

Les question peuvent être traitées de manière indépendante. Prenez le temps de lire le sujet jusqu'au bout. Le problème porte sur l'implémentation du jeu 'Pickomino'. Il s'agit d'un jeu de dés.
Selon son résultat aux dés, chaque joueur accumule des tuiles rapportant des points, mais il peut aussi se les faire voler.

Au départ, les tuiles sont disposées dans un pot commun. Il y en a 16, numérotées de 21 à 36. A chaque manche, une succession de lancers permet de définir un score qui détermine quelle tuile peut prendre le joueur. Le joueur peut, au choix, prendre une tuile de score inférieur ou égal dans le pot commun, ou une tuile de valeur strictement égale chez un autre joueur si c'est la dernière que cet autre joueur a obtenu.

Le jeu se termine lorsqu'il ne reste plus de tuiles dans le pot commun, ce qui signifie que le pot commun va toujours contenir au moins une tuile.

Les tuiles seront stockées dans un tableau, mais seront chaînées pour permettre de mémoriser leur position. Elles appartiendront donc à une liste chaînée ou une autre selon leur position en jeu. Cela signifie qu'il faut les retirer d'une liste avant des les inclure dans une autre liste.

a) écrire la structure de données **`tuile`** représentant une tuile. Cette structure contient les champs suivants :
- `numero` : le numéro de la tuile (ce numéro est unique)
- `points` :le nombre de points que rapporte la tuile
- `suivant` : un pointeur vers une autre structure **`tuile`**

b) écrire la déclaration du tableau contenant les 16 tuiles

c) La table suivante indique les points que rapporte chaque tuile. Proposez une formule qui permet de déduire ce nombre de points depuis le numéro de la tuile.

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4

d) au départ, les tuiles sont organisées dans une seule liste croissante. Cela signifie que le champ **`suivant`** de la tuile 21 pointe vers la tuile 22 et ainsi de suite.

Quelle doit être la valeur du champ **`suivant`** de la tuile 36 ?

e) écrire la fonction **`init_tuiles`** qui initialise les valeurs des tuiles. Cette fonction reçoit un seul argument : le tableau déclaré à la question b. Le nombre de cases est déjà connu : 16. La fonction ne retourne pas de résultat car elle modifie directement le tableau reçu en argument.

f) écrire la structure de données **`joueur`** qui contient les champs suivants :
- `nom` : le nom ou pseudonyme du joueur concerné
- `tuiles` : un pointeur vers une tuile. Ce sera la dernière tuile que le joueur aura acquise. Plus la tuile sera ancienne, plus elle sera éloignée dans la liste.

g) écrire la fonction **`init_joueur`** qui crée puis renvoie une structure **`joueur`** en demandant son nom à l'utilisateur du programe. Cette fonction ne reçoit aucun argument.
Au départ, les joueurs ne possèdent pas de tuile.

h) écrire la fonction **`cree_joueurs`** qui reçoit un nombre de joueurs en argument et qui crée un tableau contenant autant de structures **`joueur`** que nécessaire (elles seront initialisées). La fonction retourne un pointeur pour utiliser ce tableau.

i) écrire la fonction **`gagne_tuile`** qui reçoit un pointeur vers une structure **`joueur`** et un pointeur vers une structure **`tuile`**. Cette fonction ajoute la nouvelle tuile parmi celles du joueur, en première position.

j) écrire la fonction **`cherche_tuile`** qui reçoit : un tableau de structures **`joueur`**, un nombre de joueurs et un numéro de tuile. Si la tuile dont le numéro est donné en argument est la première tuile d'un des joueurs, la fonction renvoie un pointeur vers la structure **`joueur`** correspondant. Sinon, la fonction renvoie **NULL**.

k) écrire la fonction **`tuile_disponible`** qui reçoit un pointeur vers une structure **`tuile`** et un numéro de tuile. Cette fonction retourne le numéro de la tuile la plus élevée qui
- est dans la liste de tuiles
- est inférieure ou égale au numéro passé en argument.

l) écrire la fonction **`vole_tuile`** qui reçoit un pointeur vers une structure **`joueur`**, qui retire la première tuile de celles du joueur (celle qu'il a acquise le plus récemment) et qui retourne un pointeur vers cette tuile.
INDICE : pour retirer la première tuile, il suffit que la structure **`joueur`** pointe directement sur la deuxième tuile.

m) Pour manipuler une liste, on envoie généralement un pointeur vers son premier élément. Cela pose problème si on veut retirer un élément de la liste, car cet élément est potentiellement le premier.
Expliquer comment contourner ce problème et/ou pourquoi cela ne s'applique pas à la question précédente.

n) écrire la fonction **`pioche_tuile`** qui reçoit un numéro de tuile et ce qui est nécessaire pour identifier une liste de tuiles. Cette fonction retire la tuile identifiée par le numéro et renvoie un pointeur vers cette même tuile.

L'implémentation du jeu n'est pas terminée car il reste à gérer les lancers de dés et l'organisation des tours de jeu. Le sujet se termine à ce niveau d'avancement.