AP1 TP3
Le Chifoumi

Lors de ce TP vous allez créer un programme permettant de jouer des parties de chifoumi. Le chifoumi est un jeu que vous connaissez sûrement sous le nom de « Pierre-Ciseaux-Feuille ». Il s'agit d'un jeu pour 2 joueurs.

Les deux joueurs choisissent simultanément un des trois coups possibles : la pierre bat les ciseaux (en les émoussant), les ciseaux battent la feuille (en la coupant), la feuille bat la pierre (en l'enveloppant). Ainsi chaque coup bat un autre coup, fait match nul contre le deuxième (son homologue) et est battu par le troisième (Cf. Wikipédia).

Votre programme permettra de gérer une partie entre deux joueurs humains ou un joueur humain face à un joueur artificiel ou 2 joueurs artificiels. Une partie se déroulera en plusieurs manches (le nombre de manches sera choisi en début de partie).

Type de données manipulées

Afin de vous aider, voici les types que nous avons définis pour vous :

```
struct  mscore_partie
{
    int nbPerdue;
    int nbGagnee;
    int nbNulle;

};
typedef struct mscore_partie  ScorePartie;
struct  mjoueur
{
    text nom;
    ScorePartie score;
    bool humain;
};
typedef struct mjoueur Joueur
struct  mpartie
{
    Joueur j1;
    Joueur j2;
    int nbManche;
    int nbMancheJouee;

};
typedef struct mpartie  Partie;
```

Une Partie est définie par 2 Joueur, un nombre de manche et le nombre de manche actuellement jouées. Un Joueur est défini par son nom, son score et le fait qu'il soit humain ou non. Un ScorePartie est défini par le nombre de manche(s) gagnée(s), perdue(s) ou nulle(s).

Dans notre programme la Pierre correspondra à la valeur 1, les Ciseaux à la valeur 2 et la Feuille à la valeur 3.

Le tirage de nombre(s) aléatoire(s) en C

Dans le cas où un joueur est simulé par l'ordinateur il nous faut pouvoir tirer un nombre aléatoire (ou plutôt pseudo-aléatoire). Il est de possible, en C et dans bien d'autres langages, de faire appel à un générateur de nombres aléatoires. On initialise (srand()) le générateur (ce qui permet de d'initialiser l'indice de la séquence) avec l'heure système (time()). La fonction rand retourne un nombre entier compris entre 0 et RAND_MAX.

Voici le code C d'une fonction qui retourne une valeur pseudo-aléatoire comprise entre 2 valeurs passées en paramètre :

```
int alea (int bmin, int bmax)
{

    int nombre;


    nombre = bmin + rand() % (bmax - bmin + 1);

    return nombre;

}
```

Remarque le générateur est initialisé dans le main de la manière suivante :

```
srand(time(NULL));
```

Travail demandé

Ecrire les fonctions suivantes :

ScorePartie initScore() : cette fonction permet de retourner une valeur de type ScorePartie dont les valeurs des trois champs sont nulles.

Joueur creerJoueur(bool humain) : cette fonction permet de retourner une valeur de type Joueur, la fonction demandera le nom du joueur à l'utilisateur, le fait qu'il s'agisse d'un joueur humain ou non est passé en paramètre d'entrée.

int demanderNombreEntre(int bmin, int bmax) : cette fonction retourne un entier saisi par l'utilisateur. Néanmoins la fonction n'acceptera qu'une réponse comprise entre les deux valeurs passées en paramètre d'entrée. Tant que la saisi de l'utilisateur est incorrecte, la question lui est posée. (Vous utiliserez cette fonction chaque fois que vous le jugerez nécessaire).

Partie creerPartie() : cette fonction retourne une valeur de type Partie dont les différentes valeurs initiales ont été demandées à l'utilisateur.

int vainqueur(int j1,int j2) : cette fonction admet en paramètre d'entrée le coup du joueur 1 et le coup du second joueur. La fonction retourne 1 si c'est le joueur 1 qui gagne, 2 si c'est le joueur 2 et 0 en cas d'égalité.

Partie MAJScore(Partie p, int v) : Cette fonction reçoit en paramètre d'entrée une valeur de type Partie, et une valeur qui indique quel est le vainqueur de la manche en cours (0,1 ou 2 : même codification que la fonction précédente). La fonction met, alors, à jour le score des deux joueurs de la partie.

Partie jouerUneManche(Partie p) : cette fonction reçoit en paramètre d'entrée une valeur de type Partie. Elle demande pour les joueurs humains leur choix de coup, ou fait appel à la fonction aléa() pour les joueurs artificiels. Puis la fonction met à jour les scores après avoir déterminé le vainqueur de la manche. La partie ainsi mise à jour est retournée.

void **jouerPartie**() : cette fonction crée une nouvelle Partie, lance les différentes manches, puis affiche le nom du joueur qui a gagné la partie.

int main() : la fonction main() lancera une nouvelle partie tant que l'utilisateur le souhaite.

Pour aller plus loin:

Modifiez vos programmes pour que le nombre de valeurs en jeu (ici 3: ciseaux, pierre, feuille) puisse être choisi au début du jeu, ce nombre doit être un nombre impair. Ce jeu existe avec par exemple 9 valeurs ou 13 valeurs ou … Dans le cas où il y a n valeurs, on s'efforce de garder l'équilibre des chances, un choix doit être battu par autant de coups qu'il en bat. Ici, un choix bat un choix, est battu par un autre et fait match nul avec le troisième.

Pour 3 valeurs, on a le tableau de gagnants suivant:

Gagnants	**1**	**2**	**3**
1	nul	1	3
2	1	nul	2
3	3	2	nul

Pour 5 valeurs nous aurions:

gagnants	1	2	3	4	5
1	nul	1	3	1	5
2	1	nul	2	4	2
3	3	2	nul	3	5
4	1	4	3	nul	4
5	5	2	5	4	nul

Pour généraliser, observons que dans le cas d'un coup (a,b) en supposant a<b, alors a gagne si et seulement si b-a est impair, le coup est nul si b=a et c'est b qui gagne si b-a est pair.