Le Lycée Kléber organise cette année un tournoi de tennis en double, durant lequel des équipes de deux se font face. Chaque équipe engagée se compose de 2 joueurs ou joueuses, et possède un nom (une chaîne de caractère). Chaque joueur est caractérisé par un nom et un prénom (deux chaînes), un âge entier, et un score allant de 0 à 100 indiquant son niveau. Le niveau d'une équipe est simplement défini comme la somme des niveaux de ses deux joueurs.

Ex1. Types et affichages en OCaml - 1h

1. Définir des types enregistrement `joueur` et `equipe` pour représenter un joueur et une équipe.
2. Définir un exemple de deux joueurs dans une équipe, *par exemple* deux joueurs nommés Yannick Noah (de 61 ans) et Serena Williams (de 40 ans), réunis dans une équipe, du nom de votre choix. Vous leur choisirez un score quelconque.
3. Définir une fonction `affiche_joueur : joueur -> unit` qui affiche un joueur sous le format suivant. Penser à utiliser `Printf.printf` et des drapeaux comme en C, par exemple `Printf.printf "%s %s (%i ans) de score %i/100.\n" j.prenom j.nom j.age j.score`.

```
Yannick Noah (61 ans) de score 80/100.
```

4. Définir une fonction `affiche_equipe : equipe -> unit` qui affiche une équipe sous le format suivant. On se souviendra que pour afficher ", il faut l'échapper par `\"` dans une chaîne : `Printf.printf "Equipe \"%s\" :\n" e.nom_equipe;`

```
Equipe "Super Forte" :
- Yannick Noah (61 ans) de score 80/100.
- Serena Williams (40 ans) de score 95/100
```

5. À l'aide des fonctions `read_line : unit -> string` et `read_int : unit -> int` qui permettent de lire respectivement une chaîne et en entier depuis l'entrée standard (= le clavier), écrivez une fonction `saisir_joueur : unit -> joueur`. Tester la pour définir deux autre joueurs (ayant comme nom, prénom, âge et score des valeurs de votre choix). Puis utiliser la pour définir une autre fonction `saisir_equipe : unit -> equipe`. Si besoin, allez regarder la documentation de `read_line` et `read_int` sur la page `https://ocaml.org/api/Stdlib.html#2_Inputfunctionsonstandardinput`.
6. Pour se faire affronter deux équipes, on va simplement dire que l'équipe gagnante est celle qui a le plus grand score (= somme des scores des deux joueurs). En cas d'égalité, on fera un appel à `Random.int 2` pour déterminer aléatoirement le numéro de l'équipe gagnante (0 ou 1). Avant toute utilisation de fonctions du module `Random`, il convient d'initialiser le générateur de nombre aléatoire avec un appel à `Random.self_init()` (qui prend du "vrai" aléatoire dans l'ordinateur en écoutant des événements extérieurs comme l'activité de la souris, du trafic Internet, de la température du CPU, etc.). Écrire une fonction `simuler_match : equipe -> equipe -> int` qui fait s'affronter deux équipes et renvoie le numéro de l'équipe gagnante (1 ou 2). En utilisant `affiche_equipe`, on pensera à faire s'afficher un message pour voir quelle équipe a gagné, de la forme suivante :

```
Dans un match opposant
Equipe "Super Forte" :
```



```
- Yannick Noah (61 ans) de score 80/100.
- Serena Williams (40 ans) de score 95/100
à
Equipe "Ligue des Justiciers" :
- Bruce Wayne (29 ans) de score 99/100.
- Wonder Woman (3390 ans) de score 99/100
c'est l'équipe "Ligue des Justiciers" qui a gagné.
```

7. (bonus) On veut désormais simuler un tournoi opposant n équipes, stockées dans un `equipe array` de taille `n`. Chaque équipe rencontre toutes les autres une seule fois, et elle obtient un nombre de point égal au nombre de matchs gagnés. En initialisant un tableau de taille n stockant les scores de chaque équipe (`Array.make n 0` est votre amie), et avec deux boucles `for` imbriquées et $n(n-1)/2$ appels à la fonction `simuler_match`, calculer l'équipe gagnante du tournoi. En cas d'égalité, on prendra une équipe arbitraire parmi celles ayant obtenu un score maximal. C'est-à-dire qu'il faut écrire une fonction `simuler_tournoi : equipe array -> equipe` qui renvoie (et affiche à la fin) l'équipe gagnante.

Ex.2 Types et affichages en C - 1h

Écrire une fichier C `TP8.c` important `stdio.h`, `stdlib.h` et `time.h`.

On rappelle qu'on compile ce fichier avec `COMPILATEUR = gcc` ou `clang` avec la ligne de commande suivante, puis on exécute le binaire produit avec la dernière ligne :

```
$ COMPILATEUR -O0 -Wall -Wextra -Wvla -Werror -fsanitize=address
  -fsanitize=undefined -pedantic -std=c11 -o TP8.exe TP8.c -lm
$ ./TP8.exe
```

Si une ligne affiche un résultat qui vous semble bizarre, commenter le.

1. Définissez les types structures `joueur` et `equipe` correspondants.
2. Définir une fonction `void affiche_joueur(struct joueur j)` qui affiche à l'écran un joueur, et `void affiche_equipe(struct equipe e)` qui affiche à l'écran une équipe, au même format que dans la partie précédente. On utilisera des `printf`.
3. Écrivez une fonction `void saisir_joueur(struct joueur* j)` permettant de saisir au clavier les caractéristiques d'un joueur. Le paramètre doit être passé par adresse. On utilisera la fonction `scanf("%s", j->prenom);` ou `scanf("%i", &(*j).age);` pour lire depuis le clavier et stocker le résultat dans un des champs de la structure `joueur`. Les deux syntaxes `(*j).champ` et `j->champ` sont équivalentes.
4. Faites de même pour écrire une fonction `void saisir_equipe(struct equipe* e)`. Le paramètre doit être passé par adresse. Elle utilisera un `scanf("%s", e->nom)` pour le nom de l'équipe, et deux appels à `saisir_joueur(&e->joueur1)` et `saisir_joueur(&e->joueur2)`.
5. Écrire une fonction `main` qui crée et remplit un tableau de `4` équipes en utilisant la fonction `saisis_équipe`.
6. Compléter la fonction `main` précédente de manière à afficher les équipes après la saisie.

7. *Bonus* : terminer de faire comme dans la partie en OCaml, pour simuler un match puis un tournoi en C. On pourra utiliser `srand( (unsigned)time( NULL ) );` pour initialiser le générateur de nombres aléatoires, puis `1 + (rand() % 2)` pour obtenir un entier aléatoire uniforme entre `1` et `2`.

