**LANGAGES DE PROGRAMMATION**
**IFT 3000 NRC 54909**
**ÉTÉ 2018**

# Travail pratique 1 (individuel)

À remettre, par le portail du cours avant 17h00 le mercredi 13 juin 2018

## 1 Objectifs

Nous vous demandons dans ce travail pratique d'implanter un système de recherche de matchs des coupes du monde de soccer. Vous devez utiliser les notions que nous avons vues dans le cours en faisant des manipulations de base sur un système de recherche de matchs qui est implanté en utilisant les listes, les enregistrements et les arbres.

## 2 Mise en contexte

Nous vous proposons un travail qui traite d'un sujet d'actualité. En effet et étant donné le déroulement cet été de la coupe du monde de soccer en Russie, nous vous demandons d'utiliser les données ouvertes des différentes coupes du monde en utilisant entre autres ces trois sites : FIFA, Github et Fixturedownload.

Cependant et afin de simplifier le travail demandé ainsi que le travail de correction, nous vous demandons de ne pas télécharger directement les données disponibles sur les sites Web précédents, mais d'utiliser le fichier de données « data.csv » pré-téléchargé pour vous et disponible sur le site Web du cours. De plus et afin d'uniformiser l'affichage, un prétraitement a été fait sur ce fichier pour rassembler les données et enlever les accents ainsi que certaines irrégularités dans ces données.

Dans ce fichier, il y a exactement les données de 21 coupes du monde de 1930 à 2018. Ces coupes du monde se déroulent normalement tous les 4 ans. Il est à noter par contre que les compétitions des années 1942 et 1946 ont été annulées à cause de la guerre mondiale. Les données sont séparées par des virgules et les résultats des matchs de la coupe de cette année (2018) ne sont évidemment pas encore disponibles. Voici un exemple de matchs présents dans le fichier :

| world_cup | id | home | away | h_score | a_score | match_number | new_match_number | date | time | stadium | attendance | phase | year | host |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1990 FIFA World Cup | 264 | Italy | Uruguay | 2 | 0 | 42 | 42 | 25 June 1990 | 21:00 | Rome / Stadio Olimpico | 73303 | Round o | 1990 | Italy |
| 1990 FIFA World Cup | 181 | Spain | Yugoslavia | 1 | 2 | 43 | 43 | 26 June 1990 | 17:00 | Verona / Marc Antonio Bentegodi | 35500 | Round o | 1990 | Italy |
| 1990 FIFA World Cup | 55 | England | Belgium | 1 | 0 | 44 | 44 | 26 June 1990 | 21:00 | Bologna / Renato Dall Ara | 34520 | Round o | 1990 | Italy |
| 1990 FIFA World Cup | 31 | Yugoslavia | Argentina | 0 | 0 | 45 | 45 | 30 June 1990 | 17:00 | Florence / Comunale | 38971 | Quarter | 1990 | Italy |
| 1990 FIFA World Cup | 243 | Italy | Republic of Ir | 1 | 0 | 46 | 46 | 30 June 1990 | 21:00 | Rome / Stadio Olimpico | 73303 | Quarter | 1990 | Italy |
| 1990 FIFA World Cup | 197 | Germany FR | Czechoslovak | 1 | 0 | 47 | 47 | 01 July 1990 | 17:00 | Milan / Giuseppe Meazza | 73347 | Quarter | 1990 | Italy |
| 1990 FIFA World Cup | 103 | England | Cameroon | 3 | 2 | 48 | 48 | 01 July 1990 | 21:00 | Naples / San Paolo | 55205 | Quarter | 1990 | Italy |
| 1990 FIFA World Cup | 28 | Italy | Argentina | 1 | 1 | 49 | 49 | 03 July 1990 | 20:00 | Naples / San Paolo | 59978 | Semi-fir | 1990 | Italy |
| 1990 FIFA World Cup | 159 | Germany FR | England | 1 | 1 | 50 | 50 | 04 July 1990 | 20:00 | Turin / Juventus Stadium | 62628 | Semi-fir | 1990 | Italy |
| 1990 FIFA World Cup | 162 | Italy | England | 2 | 1 | 51 | 51 | 07 July 1990 | 20:00 | Bari / Stadio San Nicola | 51426 | Match fo | 1990 | Italy |
| 1990 FIFA World Cup | 27 | Germany FR | Argentina | 1 | 0 | 52 | 52 | 08 July 1990 | 20:00 | Rome / Stadio Olimpico | 73603 | Final | 1990 | Italy |
| 1986 FIFA World Cup | 459 | Bulgaria | Italy | 1 | 1 | 1 | 1 | 31 May 1986 | 12:00 | Mexico City / Estadio Azteca | 96000 | Group m | 1986 | Mexico |
| 1986 FIFA World Cup | 395 | Argentina | Korea Republ | 3 | 1 | 4 | 5 | 02 June 1986 | 12:00 | Mexico City / Estadio Olimpico Uni | 60000 | Group m | 1986 | Mexico |
| 1986 FIFA World Cup | 394 | Italy | Argentina | 1 | 1 | 13 | 13 | 05 June 1986 | 12:00 | Puebla / Estadio Cuauhtemoc | 32000 | Group m | 1986 | Mexico |
| 1986 FIFA World Cup | 460 | Korea Republ | Bulgaria | 1 | 1 | 14 | 15 | 05 June 1986 | 16:00 | Mexico City / Estadio Olimpico Uni | 45000 | Group m | 1986 | Mexico |
| 1986 FIFA World Cup | 643 | Korea Republ | Italy | 2 | 3 | 27 | 28 | 10 June 1986 | 12:00 | Puebla / Estadio Cuauhtemoc | 20000 | Group m | 1986 | Mexico |
| 1986 FIFA World Cup | 389 | Argentina | Bulgaria | 2 | 0 | 28 | 27 | 10 June 1986 | 12:00 | Mexico City / Estadio Olimpico Uni | 65000 | Group m | 1986 | Mexico |
| 1986 FIFA World Cup | 428 | Belgium | Mexico | 1 | 2 | 7 | 7 | 03 June 1986 | 12:00 | Mexico City / Estadio Azteca | 110000 | Group m | 1986 | Mexico |
| 1986 FIFA World Cup | 628 | Paraguay | Iraq | 1 | 0 | 10 | 10 | 04 June 1986 | 12:00 | Toluca / Bombonera - Estadio Nem | 24000 | Group m | 1986 | Mexico |
| 1986 FIFA World Cup | 680 | Mexico | Paraguay | 1 | 1 | 19 | 19 | 07 June 1986 | 12:00 | Mexico City / Estadio Azteca | 114600 | Group m | 1986 | Mexico |

Chaque match a donc un code (*id*) permettant de l'identifier. De plus, nous pouvons identifier une coupe du monde en utilisant l'année de déroulement de cette compétition sportive internationale.

De ce fait, une paire contenant le code du match et l'année de la coupe du monde permet d'identifier d'une façon unique un match donné. Nous utilisons justement cette paire comme clé dans un arbre binaire de recherche. Les données associées à ces clés seront évidemment les matchs de soccer. Cet arbre est en fait un Map (à ne pas confondre avec la fonction d'ordre supérieur map) qui est une structure de données déjà implantée dans OCaml. Les méthodes que vous pouvez appliquer sur cette structure se trouvent ici :
http://caml.inria.fr/pub/docs/manual-ocaml/libref/Map.Make.html
Nous devons utiliser un module comme paramètre d'entrée d'un foncteur afin de créer cette structure. Cela est déjà fait pour vous dans le code joint avec cet énoncé (PaireCles et SMatchsMap). Pour plus d'information, veuillez consulter ce lien : https://ocaml.org/learn/tutorials/map.html

# 3 Système de matchs

Le système de matchs de soccer est implanté en utilisant entre autres le type d'enregistrement suivant:

```
type smatch = {
    annee_coupe_monde : string;
    id_smatch : string;
    equipe1 : string;
    equipe2 : string;
    score_equipe1 : string;
    score_equipe2 : string;
    numero_smatch : string;
    date_smatch : string;
    stade_smatch : string;
    pays_organisateur: string;
```

}

Cet enregistrement contient plusieurs champs de chaînes de caractères représentant les attributs des données du fichier fourni « data.csv ». Nous utilisons également un type défini par abréviation (smatchs_coupe_monde) représentant une liste de matchs appartenant à une ou plusieurs coupes du monde.

# 4 Éléments fournis

Le TP consiste à compléter le fichier "TP1-E2018.ml" représentant la structure **SysSMatchs** qui a la signature **SYSSMATCHS** (voir fichier: "TP1-SIG-E2018.mli"). **Il est à noter qu'il ne faut pas modifier la signature SYSSMATCHS.**

Les fonctions fournies sont les suivantes (vous pouvez en ajouter au besoin):

A. Fonctions utilitaires de manipulation des listes, des chaînes de caractères et/ou des fichiers:

1. **appartient e l : 'a → 'a list → bool** est une fonction qui prend un élément *e* de type polymorphe et une liste d'éléments de même type et rend un booléen qui vaut *true* si l'élément *e* est présent dans la liste et *false* sinon.
2. **enlever e l : 'a → 'a list → 'a list** est une fonction qui prend un élément *e* de type polymorphe et une liste d'éléments de même type et qui rend une liste où *e* a été supprimé partout s'il existe.
3. **remplacer e e' l : 'a → 'a → 'a list → 'a list** est une fonction qui prend deux éléments *e* et *e'* d'un même type polymorphe et une liste d'éléments également de même type et qui rend une liste où *e* a été remplacé partout par *e'*.
4. **uniques lch : string list → string list** est une fonction qui prend une liste de chaînes de caractères *lch* et retourne une liste ne contenant que des éléments uniques. Il est à noter que les chaînes vides sont enlevées de la liste ainsi que les espaces inutiles avant et/ou après les chaînes.
5. **decouper_chaine ch : string → string → string list** est une fonction qui prend une chaîne de caractères *ch* et retourne une liste en découpant cette chaîne selon un séparateur (p.ex ",").
6. **timeRun : ('a -> 'b) →'a →'b * float** est une fonction qui permet d'estimer la durée d'exécution d'une fonction passée en argument; elle prend en argument la fonction à évaluer et un argument, et retourne le résultat de cette application ainsi que la durée de cette application.
7. **lire_fichier f s : in_channel → string → string list list** est une fonction qui prend un flux *f* et une chaîne de caractères *s* représentant un séparateur. Elle va lire un fichier texte (dans notre cas de type CSV) et retourne une liste de listes de chaînes de caractères en utilisant le séparateur pour découper les chaînes**.** Cette fonction utilise la fonction lire_ligne qui lit une ligne dans un fichier.

B. Fonctions de manipulation du système de matchs:

1. **creer_smatch lch : string list → smatch** est une fonction qui permet de retourner un match en utilisant une liste de chaînes de caractères *lch*.
2. **ajouter_smatch : string list → unit** est une fonction qui ajoute un match dans le Map.
3. **ajouter_liste_smatchs : string list list →unit** est une fonction qui ajoute une liste de matchs dans le Map en utilisant une liste de listes de chaînes de caractères.
4. **charger_donnees : string → unit** est une fonction qui permet de charger les données dans le Map.
5. **retourner_donnees : unit → smatchs_coupe_monde list** est une fonction qui permet de retourner les données du Map, mais dans une liste de listes. En fait, chaque sous-liste représente les matchs de soccer d'une coupe du monde.
6. **smatch_existe : smatch → bool** est une fonction qui retourne si un match existe dans le Map.
7. **retourner_nbr_smatchs : unit → int** est une fonction qui retourne le nombre de matchs dans le Map.

# 5 Fonctions à implanter

Votre travail consiste à implanter les 9 fonctions suivantes dont la spécification de typage est donnée dans la signature SYSSMATCHS. Il est à noter que vous n'êtes pas obligé d'utiliser toutes les fonctions fournies afin d'implanter ces fonctions demandées.

1. **retourner_smatch : string * string → smatch** est une fonction qui prend une clé contenant une paire de chaînes de caractères et retourne le match associé à cette clé dans le Map **(6 points)**.
2. **supprimer_smatch : string * string → unit** est une fonction qui permet de supprimer un smatch dans le Map et elle ne doit rien faire si le match n'existe pas **(6 points)**.
3. **supprimer_liste_smatchs : (string * string) list → unit** est une fonction qui permet de supprimer une liste de matchs dans le Map **(7 points)**.
4. **afficher_smatch : smatch → unit** est une fonction qui prend un smatch *sm* et affiche ce match à l'écran **(8 points)**. Voici un exemple d'affichage d'un match :

   Code du match: 81572532.
   Coupe du monde: Brazil 2014.
   Equipes et resultats: Mexico 1 - 0 Cameroon.
   Date: 11/06/2014.
   Stade: Estadio das Dunas Natal.

5. **afficher_smatchs_coupe_monde : smatchs_coupe_monde → unit** est une fonction qui permet d'afficher tous les matchs de la liste **(9 points)**.
6. **retourner_smatchs_coupe_monde : string → smatchs_coupe_monde** est une fonction qui permet de retourner les données d'une coupe du monde sous la forme de liste de smatch à partir du Map **(9 points).**
7. **retourner_resultats_equipe_cm : string → string → smatchs_coupe_monde** est une fonction qui permet de retourner les résultats d'une équipe dans une coupe du monde sous la forme de liste de smatch à partir du Map **(9 points).**
8. **retourner_resultats_equipe_cms : string → smatchs_coupe_monde** est une fonction qui permet de retourner les résultats d'une équipe dans toutes les coupes du monde sous la forme de liste de smatch à partir du Map **(9 points).**
9. **lancer_systeme_smatchs : string → unit** est une fonction qui permet de charger les données en utilisant le fichier csv. Elle doit chercher les informations d'une ou plusieurs coupes du monde selon le choix de l'utilisateur. Veuillez consulter le fichier « je.ml » pour voir un exemple d'exécution de cette fonction. Vous devez d'ailleurs produire les mêmes résultats en utilisant entre autres les fonctions suivantes **(25 points)** :

   *charger_donnees,*
   *timeRun,*
   *retourner_resultats_equipe_cm,*
   *retourner_resultats_equipe_cms*
   *afficher_smatchs_coupe_monde*

   Ps. pour demander à l'utilisateur une information, vous pouvez le faire par exemple de cette façon :

   ```
   flush stdout;;
   let choix = read_line () in ..
   ou
   let choix = read_int () in ..
   ```

# 6 Démarche à suivre

Puisque la structure SysSMatchs contient des fonctions qui ne sont pas encore implantées (c'est à vous de le faire), il ne vous sera pas possible de tester cette structure ou de charger le fichier « TP1-E2018.ml », avant de compléter la définition de ces fonctions. La démarche à suivre est la suivante:

- Utiliser un environnement comme Emacs, Sublime ou Eclipse, pour compléter l'implantation des fonctions.
- Une fois la structure SysSMatchs complétée, il vous sera alors possible de la tester et de charger le fichier « TP1-E2018.ml » pour y tester les fonctions qui y sont définies.
- Le test peut se faire en utilisant le fichier fourni « je.ml ». Dans ce même fichier, il y a les résultats attendus. Vous devez donc vérifier que vous obtiendrez exactement les mêmes résultats.
- Il faut donc mettre ces deux fichiers ainsi que « TP1-SIG-E2018.mli » et « data.csv » dans un même répertoire et d'exécuter la commande suivante: #use "je.ml";; (il ne faut pas oublier le # avec la commande use).

# 7 À remettre

Vous devez rendre un fichier .zip comportant uniquement les fichiers suivants :

- Le fichier « TP1-E2018.ml » complété.
- Les fichiers « TP1-SIG-E2018.mli », « je.ml » et « data.csv » non modifiés.
- Le fichier « NoteTp1.xls (un fichier Excel contenant le barème de correction. Il faut juste ajouter votre nom et matricule sur la première ligne).

Le nom du .zip doit respecter le format suivant : TP1-Matricule.zip. Nous vous rappelons qu'il est important de faire la remise par voie électronique uniquement en vous connectant à: http://monportail.ulaval.ca/ (aucun travail envoyé par courriel n'est accepté). Il est toujours possible de faire plusieurs remises pour le même travail. Vous pouvez donc remettre votre travail plus tôt afin d'être sûr qu'il a été remis en temps, et remettre par la suite une version corrigée avant l'heure limite de remise. De plus, il est de votre responsabilité de vérifier après la remise que vous nous avez envoyé les bons fichiers (**non vides et non corrompus**), sinon vous pouvez avoir un zéro.

**Attention !** Tout travail remis en retard se verra pénalisé de -25% de la note. Le retard débute dès la limite de remise dépassée (dès la première minute). Un retard excédant une journée provoquera le rejet du travail pour la correction et la note de 0 pour ce travail. Il est à noter que **12 points** sont donnés pour le respect et la qualité des biens livrables ainsi que pour la structure générale du code. (commentaires, indentation, compilation sans warnings, etc.). De plus, votre code doit absolument pouvoir être exécuté sans erreurs sur la machine virtuelle du cours. Si vous ne testez pas suffisamment votre travail, il risque de provoquer des erreurs à l'exécution lors de la correction. La moindre erreur d'exécution rend la correction extrêmement difficile et vous en serez donc fortement pénalisés.

**Bon travail !**