

## BUT1 – SAÈ S1.02

### COMPARAISON D'APPROCHES ALGORITHMIQUES

#### ‘Crazy Circus’, un jeu de Dominique Ehrhard

(<http://crazycircus.free.fr/>)

## 1 Présentation du jeu

Le jeu a pour but de déterminer quel est le meilleur dompteur parmi les joueurs. Il s'agit de réussir un exercice de domptage mettant en œuvre trois animaux : un lion, un ours blanc et un éléphant. Ces animaux sont répartis sur deux podiums – l'un est bleu, l'autre est rouge. Lorsque deux ou trois animaux sont sur le même podium, ils forment une pile (au sens commun du terme) comme cela est illustré à la fin de ce document.

Les animaux réagissent uniquement à 5 ordres distincts :

<b>KI</b>	L'animal se trouvant en haut de la pile du podium bleu saute pour rejoindre le sommet de la pile du podium rouge
<b>LO</b>	L'animal se trouvant en haut de la pile du podium rouge saute pour rejoindre le sommet de la pile du podium bleu
<b>SO</b>	Les deux animaux se trouvant au sommet des piles des deux podiums échangent leur place
<b>NI</b>	L'animal se trouvant en bas de la pile du podium bleu monte et se place en haut de la pile de ce même podium
<b>MA</b>	L'animal se trouvant en bas de la pile du podium rouge monte et se place en haut de la pile de ce même podium

Le but du jeu est de trouver le plus rapidement la bonne séquence d'ordres qui, réalisés par les animaux, les conduiront d'une position de départ à une position à atteindre. Le joueur ayant trouvé le premier une bonne séquence remporte le tour. Un joueur ayant donné une mauvaise séquence ne peut plus en proposer d'autre pour ce tour. Si, durant un tour, il n'y a plus qu'un joueur à ne pas avoir proposé de séquence, celui-ci remporte le tour.

La position initiale de début du jeu et la position à atteindre à chaque tour sont déterminées en tirant au hasard l'une des 24 cartes objectif représentant chaque position possible. La position atteinte après un tour est la position de départ du tour qui suit. Une carte une fois tirée n'est pas remise en jeu et la partie se termine lorsque les cartes sont épuisées. Le joueur ayant remporté le plus de tours gagne la partie.

Les règles complètes du jeu original sont accessibles en suivant ce lien : [http://crazycircus.free.fr/goodies/regles\\_crazy\\_circus.pdf](http://crazycircus.free.fr/goodies/regles_crazy_circus.pdf)

## 2 Travail à faire

Vous devez réaliser un programme permettant à des joueurs de s'affronter. Votre programme devra gérer une partie dans sa totalité.

Les noms des animaux impliqués dans le jeu ainsi que les ordres autorisés durant la partie seront lus à partir du fichier nommé **crazy.cfg** se trouvant dans le répertoire courant. Ce fichier contient deux lignes. La première liste le nom des animaux séparés par un espace et la seconde les ordres choisis (parmi les 5 possibles). Par exemple, le fichier **crazy.cfg** d'une partie classique contiendra :

```
OURS ELEPHANT LION  
KI LO SO NI MA
```

Pour qu'un tel fichier soit valide, les animaux doivent être deux au minimum. Les ordres sont nécessairement choisis parmi les 5 ordres connus et doivent être au moins trois. Si ce n'est pas le cas, votre programme doit se terminer immédiatement après avoir affiché un message informatif (sur une ligne).

Si durant un tour du jeu, les ordres autorisés ne permettent pas d'atteindre la position donnée par la carte objectif, votre programme n'a pas à détecter que le tour n'est pas jouable. Les séquences d'ordre jouées seront toutes invalides et le dernier joueur encore en jeu remportera le tour.

Les identités des joueurs (leurs prénoms ou surnom par exemple) doivent être reçus sur la ligne de commande au lancement du programme. Les joueurs doivent être 2 au minimum et avoir des noms distincts. Si ce n'est pas le cas, votre programme doit se terminer immédiatement après avoir affiché un message informatif (sur une ligne).

Votre programme doit commencer par rappeler aux joueurs les différents ordres possibles en affichant la ligne qui suit (en vous limitant aux ordres autorisés par `crazy.cfg`).

KI (B -> R) | LO (B <- R) | SO (B <-> R) | NI (B ^) | MA (R ^)

Les sauts d'un sommet à l'autre sont représentés par des flèches et un déplacement du bas vers le haut est symbolisé par le caractère `^`. La lettre B désigne le podium bleu et R le rouge.

Votre programme doit ensuite afficher la situation initiale du jeu sous la forme reproduite ci-dessous.

```
OURS          LION
LION ELEPHANT ELEPHANT OURS
---- ---- ==> ---- ----
BLEU ROUGE    BLEU  ROUGE
```

La position de départ est donnée en haut à gauche (ici le lion est sous l'ours sur le podium bleu alors que l'éléphant est seul sur le podium rouge) et la position à atteindre est indiquée à droite (ici le lion est sur l'éléphant sur le podium bleu alors que l'ours est seul sur le podium rouge). Les séquences KIMALONI et SONI sont deux solutions pour cette situation de jeu.

Lorsqu'un joueur veut jouer, il saisi son identité suivie de la séquence d'ordres qu'il propose. Si DP est une identité connue, le joueur peut saisir 'DP KIMALONI' par exemple.

Si l'identité n'est pas connue, que le joueur n'avait pas le droit de jouer (*i.e.* il a déjà fait une erreur) ou que la séquence n'est pas la bonne, un message informatif (sur une ligne) doit être affiché par le programme.

Si la séquence est correcte, le joueur remporte un point, une nouvelle position de jeu est choisie aléatoirement par le programme parmi celles qui n'ont pas encore été jouées et un nouveau tour peut débiter. De même, si la séquence est incorrecte et qu'il ne reste plus qu'un joueur pouvant jouer, ce dernier remporte un point et un nouveau tour débute. Dans ces deux cas, votre programme doit afficher un message informatif (sur une ligne) avant d'afficher la nouvelle situation de jeu.

En fin de partie, le score et le rang des joueurs sont affichés (par score décroissant et par ordre alphabétique en cas d'égalité).

Seuls les formats des situations de jeu et des scores de fin de partie sont stricts. En particulier, il vous est demandé de respecter l'alignement des animaux sur leur podium. Le contenu des messages informatifs sont libres (mais doivent tous être composés d'une unique ligne de texte). Des exemples sont donnés en annexe.

### 3 Qui, quoi et quand?

Votre projet doit être fait en binôme. Les groupes de 3 ne seront pas acceptés. Évitez de faire votre projet tout seul (soit vous êtes très fort et des personnes ont besoin de votre aide, soit vous avez des difficultés et il faut vous faire aider).

Avant le **vendredi 9 janvier 2026 à 23h59**, vous devez déposer sur MOODLE une archive (.zip) contenant votre rapport (.pdf) ainsi que tous les fichiers sources (.c et .h) de votre application. Ces derniers doivent être réunis, au sein de l'archive, dans un dossier nommé **src**.

Vous devez porter une attention particulière à la rédaction de votre dossier. Sa qualité est déterminante pour l'évaluation de votre travail. Votre dossier doit être un unique document pdf dont la composition est la suivante :

- Une page de garde indiquant le nom et **le groupe** des membres du binôme, l'objet du dossier.
- Une table des matières de l'ensemble du dossier.
- Une brève introduction du projet.
- Le graphe de dépendance des fichiers sources de vos applications (*cf.* cours n°8). Tous les composants (qu'ils soient réutilisés ou développés) de vos applications devront figurer sur le graphe.
- Le code source des tests unitaires que vous aurez écrits (précisez quels tests passent et lesquels échouent).
- Un bilan du projet (les difficultés rencontrées, ce qui est réussi, ce qui peut être amélioré).
- En annexe, le code complet de vos sources (pensez à présenter les fichiers selon un ordre logique).

Nous vous rappelons que le critère principal de notation est la structuration de votre code. Votre rapport doit mettre en avant la qualité de celle-ci.

Tous les éléments (constantes, types et fonctions) déclarés au sein d'un fichier .h doivent être documentés.

Vous pouvez employer les structures de données vues en cours ou développer les vôtres. Vous pouvez vous servir de toute la bibliothèque standard du C.

La correction de votre code sera testé automatiquement par nos soins. Votre programme doit respecter strictement les formats d'affichage et de saisie présentés ci-dessus sous peine d'échouer lors des tests.

## Annexe

Ci-dessous est reproduit un extrait d'une partie classique. Trois joueurs s'y affrontent et leurs identités (JR, MZ et DP) sont fournies en paramètre de la ligne de commande. Le texte saisi par les joueurs est indiqué en **rouge**, les messages informatifs affichés par le programme sont présentés en **vert**. Le texte en **bleu** doit être affiché par votre programme en respectant scrupuleusement le même format.

```
C:> crazy.exe JR MZ DP
```

```
KI (B -> R) | LO (B <- R) | SO (B <-> R) | NI (B ^) | MA (R ^)
```

```
OURS          LION
LION  ELEPHANT  ELEPHANT  OURS
----  ----    ==>  ----  ----
BLEU  ROUGE     BLEU   ROUGE
```

```
MZ KILI
```

```
L'ordre LI n'existe pas
```

```
MZ KIKI
```

```
La sequence ne conduit pas à la situation attendue -- MZ ne peut plus jouer durant ce tour
```

```
MZ KIMALONI
```

```
MZ ne peut pas jouer
```

```
DP KIMALONI
```

```
DP gagne un point
```

```
          OURS
LION      ELEPHANT
ELEPHANT  OURS
----  ----    ==>  ----  ----
BLEU      ROUGE    BLEU  ROUGE
```

```
DP KIKIKI
```

```
La sequence ne conduit pas à la situation attendue -- DP ne peut plus jouer durant ce tour
```

```
MZ KIKINI
```

```
La sequence ne conduit pas à la situation attendue -- MZ ne peut plus jouer durant ce tour
JR gagne un point car lui seul peut encore jouer durant ce tour
```

```
      OURS
      ELEPHANT
      LION
----  ----    ==>  OURS  ELEPHANT
BLEU  ROUGE     BLEU  ROUGE
```

```
JR LOMA
```

```
JR gagne un point
```

```
...
```

```
JR MAMA
```

```
JR gagne un point
```

```
JR 21
```

```
DP 1
```

```
MZ 1
```

