
PROJET Caml : JEUX DE CARTES

Sujet à finir pour le jeudi 8 janvier 2015 23h59
À rendre par mail à aurelie.lagoutte@ens-lyon.fr

1 Consignes

Forme Vous devez rendre le fichier `.ml` contenant le code de votre projet, ainsi que le fichier exécutable. **Le fichier `.ml` devra impérativement compiler normalement, sans renvoyer d'erreurs ni de warnings.** Il sera accompagné d'un petit rapport (quelques pages) expliquant vos choix techniques, des exemples, et décrivant les difficultés que vous avez rencontrées. Le rapport sera obligatoirement un fichier `.pdf` généré en Latex. La séance de la seconde semaine de la rentrée (date à préciser) sera consacré à une séance de questions sur votre projet (environ 10 minutes chacun).

Comment lire ce sujet Ce sujet n'est *pas* un sujet de TP. C'est un sujet de projet. Il décrit la problématique et les solutions attendues. Il ne décrit pas toujours en détails les fonctions attendues une par une (sauf les parties 3 et 4), c'est à vous de faire les choix techniques, de décomposer intelligemment en petites fonctions, et de choisir l'ordre le plus adéquat pour implémenter les différentes fonctionnalités. Attention cependant pour les parties 3 et 4 : ici, on attend que vous suiviez la décomposition en fonctions et le typage de chaque fonction, à moins que vous fournissiez une solution meilleure ou plus adaptée : dans ce cas, une explication convaincante est attendue dans le rapport.

De plus, à certains endroits, plusieurs variantes sont proposées : commencez par d'abord par choisir les variantes les plus simples, et lorsque vous aurez terminé la majeure partie du sujet, affinez votre projet en choisissant des variantes un peu plus compliquées. Une partie de la note dépendra de la difficulté des variantes choisies. Vous devez expliquer dans le rapport quelles variantes vous avez choisi. Vous êtes également libres de rajouter toute fonctionnalité qui vous paraîtrait adéquate.

Enfin, vous êtes encouragés à vous servir de la documentation en ligne de Caml.

Consignes pour le code

- Évidemment, votre code devra être accompagné d'exemples judicieux et de quelques commentaires (même si la plupart des explications pourront être reportées dans le rapport).
- Pour des raisons de compatibilité, il vous est demandé de ne pas utiliser d'accents dans votre code (même dans les impressions Terminal), et de ne pas mettre d'espace dans le nom de votre fichier `.ml`.
- Bien que les TP vous aient montré l'usage de l'impératif en Caml, vous utiliserez autant que possible la programmation récursive pour vos fonctions (les fonctions d'affichages seront un mix de récursif et d'impératif) : seule la partie 1 et les appels à `print_` pourront venir dans un certaine mesure du monde impératif. Et normalement dans ce sujet, on peut se passer complètement de `while`!!
- Vous pouvez utiliser ou non les modules en Caml, ce n'est pas obligatoire.
- Tous vos filtrages devront être exhaustifs (penser aux exceptions pour les cas d'erreurs).
- Le filtrage permettra de se passer des fonctions `head` et `tail`, ainsi que de tout autre appel de fonction inutile, comme `first` pour avoir le premier élément d'un couple...

En cas de difficultés Les séances de TP sont là pour vous permettre d'avancer dans votre projet, mais aussi d'échanger avec moi sur les points qui posent problème. Je vous conseille de travailler régulièrement chez vous pour ne pas attendre la veille de la date limite.

Notation Les critères suivants entreront en compte dans la notation :

- Résolution des problématiques de l'énoncé
- Efficacité du code
- Style de programmation : fonctionnel et récursif
- Difficulté des variantes choisies, des fonctions de triche et de "l'intelligence artificielle" pour la partie 5.
- Lisibilité du code (noms de variables, indentation, commentaires)
- Qualité de l'interface avec le terminal (voir partie 2)
- Qualité du rapport
- Séance de questions/réponses

Résumé du sujet Le but de ce projet est de fournir une sorte de "Game Center" dans le terminal, proposant un jeu de Blackjack, un jeu de Menteur et un jeu de Bataille.

2 Interfacer avec le terminal

On rappelle la commande suivante qui permet de créer un fichier exécutable `jeu` à partir du fichier `projet.ml` :

```
ocamlc -o jeu projet.ml.
```

On lancera l'exécutable grâce à l'appel `./jeu`.

Il s'agit tout d'abord de créer un type `joueur` qui pourra regrouper les informations suivantes : un prénom, un crédit (nombre de jetons), et un nombre de parties gagnées. A l'exécution de `jeu`, le terminal doit afficher un message de bienvenue et demander le prénom du joueur. Le joueur commencera avec un seul jeton.

Le terminal doit ensuite afficher le menu principal (qui reviendra à chaque fin de partie après pression sur la touche Entrée) qui affiche le prénom, le crédit du joueur, et le nombre de parties de bataille gagnées. Il lui propose ensuite les choix suivants :

1. Jouer au blackjack
2. Jouer au menteur
3. Jouer à la bataille
4. Quitter

Le choix 1 doit lancer un jeu de blackjack, voir partie 4, et le choix 2 doit lancer un jeu de Menteur, voir partie 5. La réussite à un jeu de blackjack ou de menteur fait gagner un jeton au joueur.

Le choix 3 doit lancer un jeu de bataille, voir partie 6. Une partie de bataille consomme un jeton¹. Un message d'erreur doit s'afficher si le joueur demande à jouer à la bataille alors qu'il n'a plus de jetons, puis le menu principal doit de nouveau s'afficher.

Le choix 4 permet de quitter le programme, en saluant le joueur par son prénom.

1. Variante : Une victoire à la bataille rembourse le jeton consommé

Important : tout doit être programmé dans une (ou plusieurs) **fonction(s)**, puis le fichier `.ml` doit contenir un appel à la fonction principale : `lance_programme();;`.

3 Avant de jouer....

Commençons par définir un type `couleur`, contenant 4 constantes (`Coeur`, `Carreau`, `Trefle`, `Pique`), puis le type `carte`, constitué d'un entier de 1 à 13 (1 pour l'As, 11 pour le Valet, 12 pour la Dame, 13 pour le Roi), et d'une couleur. On prendra soin de définir ce type de manière à pouvoir ensuite utiliser intensivement le filtrage.

Vous écrirez :

1. Une fonction `valide: carte -> boolean` qui vérifiera qu'une carte est valide, c'est-à-dire que l'entier qu'elle contient est bien entre 1 et 13.
2. Une fonction d'affichage d'une liste de cartes `affiche_liste_cartes: carte list -> unit`. Elle écrira par exemple "As de Coeur, 10 de Pique, Roi de Trefle".
3. Une fonction `genere_jeu: unit -> carte list` qui crée un jeu de 52 cartes.
4. Une fonction `genere_mini_jeu: int -> carte list` telle que `genere_mini_jeu n` crée un jeu de cartes contenant uniquement les cartes de Coeur et de Pique de `n` à 13. (Permet de tester la suite sur un jeu plus petit, par exemple avec 10 ou 12 cartes).
5. Une fonction `melanger: carte list-> carte list` qui mélange une liste de cartes donnée². (La complexité attendue pour cette fonction est en $\mathcal{O}(n^2)$).
6. Une fonction `distribue: carte list-> carte list * carte list` qui distribue les cartes une par une pour former deux tas de cartes de même taille (à 1 près).
7. Une fonction `piocher: int -> carte list-> (carte list * carte list)` telle que `piocher n la_pioche` permet de piocher `n` cartes dans `la_pioche` et renvoie la liste des `n` cartes piochées ainsi que la pioche restante.
8. Une fonction `empiler_cartes: carte list -> carte list -> carte list` telle que `empiler_cartes mes_cartes tas` ajoute `mes_cartes` à la fin de `tas`.

4 Blackjack

Une partie de blackjack va opposer le joueur à la banque. La banque commence par tirer une carte et va ensuite en donner deux au joueur. Le but du jeu consiste à approcher ou faire le chiffre 21 sans le dépasser. La valeur des cartes est établie comme suit : les cartes de 2 à 10 conservent leurs valeurs, les figures valent 10 et l'As vaut 1.

Le joueur peut demander autant de cartes qu'il le souhaite. S'il dépasse 21, il perd. S'il s'arrête avant, la banque tire des cartes jusqu'à dépasser 17 pour ne pas prendre de risque. Si la banque fait moins que le joueur, ou qu'elle dépasse 21, le joueur gagne. S'il fait moins que la banque, il perd. Sinon, il y a égalité.

1. Créez un type `plateau_blackjack` contenant trois listes de cartes : l'une pour les cartes du joueur, la deuxième pour les cartes de la banque, et la dernière pour la pioche. *On n'utilisera pas d'enregistrement pour définir ce type.*
2. Écrivez la fonction `creer_blackjack: unit -> plateau_blackjack` qui initialise la partie : toutes les cartes sont dans la pioche, le joueur et la banque n'en ont aucune.

2. On se souviendra de l'existence de `Random.self_init()` avant d'utiliser `Random.int`

3. Écrivez la fonction `banque_pioche: int -> plateau_blackjack-> plateau_blackjack` qui prend en argument un nombre de cartes `n` et un plateau de blackjack `p` et qui renvoie le plateau de blackjack obtenu à partir de `p` après que la banque a pioché `n` cartes.
4. Même question avec la fonction `joueur_pioche: int -> plateau_blackjack -> plateau_blackjack`.
5. Écrivez une fonction `total_cartes: carte list-> int` qui renvoie le nombre de points d'une liste de cartes³.
6. Écrivez une fonction `faire_jouer_banque: plateau_blackjack->plateau_blackjack` qui fait jouer la banque jusqu'à ce qu'elle atteigne au moins un total de 17 points.
7. Écrivez la fonction `afficher_jeu: plateau_blackjack->unit` qui permet d'afficher l'état du jeu (cartes de la banque, cartes du joueur, total actuel des cartes pour chacun).
8. Écrivez la fonction `faire_jouer_joueur: plateau_blackjack->plateau_blackjack` qui pioche les cartes une par une après accord du joueur, et s'arrête lorsque le joueur décide de ne plus prendre de risques.
9. Écrivez la fonction `jouer_blackjack joueur-> unit` qui permet de jouer au blackjack.

5 menteur

Dans notre cadre, on supposera que le jeu de menteur se joue à 4 joueurs sur un jeu de 52 cartes. Voilà comment se déroule une manche : le gagnant de la manche précédente (un joueur arbitraire pour la première manche) commence à jouer en posant une carte face *visible* sur la table. Le joueur situé à sa droite autour de la table pose alors une carte face cachée en prétendant poser une carte de la même couleur. Les joueurs continuent à poser un par un une carte face cachée, jusqu'à ce que l'un des joueurs (le Dénonciateur) accuse le dernier à avoir joué (l'Accusé) de mentir sur la couleur de sa carte. On retourne alors la dernière carte posée : si elle est bien de la même couleur que la première carte, le gagnant de la manche est l'Accusé, et le perdant est le Dénonciateur. Sinon, c'est le contraire. Le perdant de la manche récupère toutes les cartes sur la table et les mets au fond de sa pile. Le but du jeu est de se débarrasser de toutes ses cartes.

Implémenter ce jeu pour pouvoir jouer avec 3 joueurs "Ordinateur" et un joueur réel, qui choisira sa carte à poser en communiquant via le terminal.

Versión simple Pour l'ordinateur, vous implémenterez tout d'abord la stratégie naïve qui consiste à poser des cartes de la couleur demandée tant que c'est possible, puis des cartes de l'autre couleur, et qui décide à chaque tour d'accuser ou non le joueur en cours avec probabilité $1/2$.

Versión plus compliquée Vous implémenterez également une stratégie plus élaborée. Libre à vous de la choisir, et expliquez pourquoi.

6 Bataille

La bataille est un jeu de cartes à deux joueurs dont les règles sont très simples : au début du jeu, les cartes sont distribuées en deux tas de même taille, un pour chaque joueur. Les joueurs ne peuvent pas regarder les cartes en leur possession ni changer l'ordre des cartes. À chaque tour de jeu, les joueurs retournent simultanément la carte sur le dessus de leur paquet. Si la carte du joueur 1 (resp. joueur 2) a une valeur plus élevée que l'autre (dans l'ordre, de la plus élevée

3. Variante : dans le vrai jeu du blackjack, chaque As peut prendre la valeur 1 ou 11, de la manière la plus favorable pour la personne (joueur ou banque) qui a pioché l'As.

à la moins élevée : As, Roi, Dame, Valet, 10, ..., 2), le joueur 1 (resp. joueur 2) gagne les 2 cartes et les remet à la fin de son paquet. Si les deux cartes ont la même valeur, il y a bataille : chaque joueur superpose une carte face cachée sur sa propre carte, puis de nouveau une carte face visible. Ce nouveau "combat" de cartes décide qui récupère les 6 cartes en jeu. Il peut y avoir éventuellement une nouvelle bataille, dans ce cas on recommence avec une carte face cachée, etc... La partie se termine lorsqu'un des deux joueurs possède toutes les cartes. Pour éviter que les parties ne bouclent, on mélangera les cartes avant de les remettre sous le paquet.

Cas particulier : Supposons que, lors d'une bataille, le joueur 1 n'ait plus assez de cartes pour finir le tour (c'est-à-dire qu'il ne lui restait plus qu'une ou deux cartes avant de commencer le tour qui a déclenché la bataille). Dans ce cas, plusieurs variantes sont possibles :

- Le joueur 2 est déclaré gagnant.
- On déclare les joueurs *ex aequo*.
- (*plus difficile*) Le joueur 1 doit piocher dans le jeu du joueur 2 (sur le dessus ou bien au hasard au milieu, selon les variantes) pour terminer la bataille, et peut donc récupérer des cartes s'il gagne la bataille. Dans ce cas, la partie continue.

Vous préciserez clairement quelle règle vous avez choisi.

Écrivez une fonction `joue_bataille: carte list * carte list -> joueur ->unit` telle que `joue_bataille (jeu1, jeu2) joueur1` affiche le déroulement d'une partie de bataille entre `joueur1` et l'ordinateur, avec les jeux de départ respectivement `jeu1` et `jeu2`. Un exemple type de résultat attendu est décrit en Annexe. En particulier, on prendra soin d'appeler `joueur1` par son prénom, de mettre en valeur les cas de bataille, et de préciser le vainqueur de la partie.

Note : Lorsque l'utilisateur choisit 2. **Jouer à la Bataille** dans l'interface avec le terminal, le programme lancera une partie de bataille sur un nombre de cartes raisonnable (la partie doit se terminer en moins de 10 secondes environ) ⁴.

7 Trichons...

Écrivez une ou plusieurs fonctions permettant de tricher ⁵ à la bataille, et/ou au blackjack/menteur. Par exemple :

- Dans le menu principal, le joueur gagne un jeton sans jouer au blackjack.
- Lors de la distribution des cartes, le joueur réussit à échapper aux cartes de faible valeur (ou au moins une partie). Attention, pour ne pas se faire prendre, il doit néanmoins faire en sorte de distribuer les cartes en deux tas de taille sensiblement égales.
- Lors de la distribution, le joueur fait en sorte d'obtenir au moins deux As.
- Lors d'une bataille, de peur de perdre une carte de forte valeur "en sandwich" (As ou Roi par exemple), le joueur peut discrètement poser la deuxième ou troisième carte de son tas, à la place de la première (uniquement si cela lui est au moins aussi favorable).
- etc...

C'est à vous de décider de quelle manière la triche sera mise en place : choix supplémentaire 0. **Jouer avec triche** dans le menu, ou bien entrer un code secret de triche dans le menu principal au lieu de taper 1, 2 ou 3, etc...

4. Variante : l'utilisateur peut lui-même choisir le nombre de cartes dans le jeu, ou plus précisément la valeur de la plus petite carte.

5. Évidemment, ceci n'est pas une incitation à la triche. Il s'agit d'un cadre purement théorique. Je décline toute responsabilité en cas de triche de la part de mes étudiants aux partiels/examens/DM...

8 Annexe

Voici l'exemple-type d'affichage d'une partie de bataille, correspondant à l'appel :

```
joue_bataille(distribue(melanger(genere_mini_jeu 9))) toto;;
```

```
Tour:10 de Pique, 9 de Coeur,
==Toto: 4 cartes; Adversaire: 4 cartes;En jeu: 2 cartes==
Toto gagne les cartes
Tour:Dame de Pique, Roi de Coeur,
==Toto: 5 cartes; Adversaire: 3 cartes;En jeu: 2 cartes==
Adversaire gagne les cartes
Tour:9 de Pique, Valet de Coeur,
==Toto: 4 cartes; Adversaire: 4 cartes;En jeu: 2 cartes==
Adversaire gagne les cartes
Tour:10 de Coeur, Dame de Coeur,
==Toto: 3 cartes; Adversaire: 5 cartes;En jeu: 2 cartes==
Adversaire gagne les cartes
Tour:Roi de Pique, Valet de Pique,
==Toto: 2 cartes; Adversaire: 6 cartes;En jeu: 2 cartes==
Toto gagne les cartes
Tour:10 de Pique, Dame de Pique,
==Toto: 3 cartes; Adversaire: 5 cartes;En jeu: 2 cartes==
Adversaire gagne les cartes
Tour:9 de Coeur, Roi de Coeur,
==Toto: 2 cartes; Adversaire: 6 cartes;En jeu: 2 cartes==
Adversaire gagne les cartes
Tour:Roi de Pique, 9 de Pique,
==Toto: 1 cartes; Adversaire: 7 cartes;En jeu: 2 cartes==
Toto gagne les cartes
Tour:Valet de Pique, Valet de Coeur,
==Toto: 2 cartes; Adversaire: 6 cartes;En jeu: 2 cartes==
BATAILLE!!
Tour:9 de Pique, Dame de Coeur,
==Toto: 0 cartes; Adversaire: 4 cartes;En jeu: 6 cartes==
Adversaire gagne les cartes
Adversaire gagne!
```

ou une version assez sympa réalisée par l'un de vos camarades pendant la première partie du semestre :

```
*****
*          Bataille          *
*****
Jeu de carte de X à As, X = ? 7
Aurélie ||||| | Adversaire 9 de Coeur VS Valet de Pique
Aurélie ||||| | Adversaire 10 de Coeur VS Valet de Coeur
Aurélie ||||| | Adversaire 7 de Pique VS 8 de Pique
Aurélie ||||| | Adversaire 8 de Coeur VS 7 de Coeur
Aurélie ||||| | Adversaire As de Coeur VS Dame de Coeur
Aurélie ||||| | Adversaire As de Pique VS Roi de Pique
Aurélie ||||| | Adversaire 10 de Pique VS Dame de Pique
Aurélie ||||| | Adversaire 9 de Pique VS Roi de Coeur
Aurélie ||||| | Adversaire 8 de Coeur VS 9 de Coeur
Aurélie ||||| | Adversaire 7 de Coeur VS Valet de Pique
Aurélie ||||| | Adversaire Dame de Coeur VS 10 de Coeur
Aurélie ||||| | Adversaire As de Coeur VS Valet de Coeur
Aurélie ||||| | Adversaire As de Pique VS 8 de Pique
Aurélie ||||| | Adversaire Roi de Pique VS 7 de Pique
Aurélie ||||| | Adversaire Dame de Coeur VS Dame de Pique BATAILLE !
Aurélie ||||| | Adversaire Valet de Coeur VS Roi de Coeur
Aurélie ||||| | Adversaire As de Coeur VS 9 de Pique
Aurélie ||||| | Adversaire 8 de Pique VS 8 de Coeur BATAILLE !
Aurélie ||||| | Adversaire Roi de Pique VS Valet de Pique
Aurélie ||||| | Adversaire 7 de Pique VS 7 de Coeur BATAILLE !
Aurélie ||||| | Adversaire 9 de Pique VS Roi de Coeur
Aurélie ||||| | Adversaire 9 de Coeur VS Valet de Coeur
Aurélie ||||| | Adversaire 8 de Pique VS Dame de Pique
Aurélie ||||| | Adversaire 8 de Coeur VS Dame de Coeur
Aurélie ||||| | Adversaire As de Pique VS 10 de Coeur
Aurélie ||||| | Adversaire Valet de Pique VS As de Coeur
Aurélie ||||| | Adversaire Roi de Pique VS Roi de Coeur BATAILLE !
Aurélie ||||| | Adversaire 10 de Coeur VS 7 de Pique
Aurélie ||||| | Adversaire 7 de Pique VS 7 de Coeur BATAILLE !
Aurélie ||||| | Adversaire Roi de Coeur VS 9 de Coeur
Aurélie ||||| | Adversaire As de Pique VS Valet de Coeur
Aurélie ||||| | Adversaire 10 de Pique VS 8 de Pique
```