IN 101 - TD 04 **Énoncé** ENSTA IP PARIS

1 Magie or not magie

≪Déterminez s'il y a un truc caché (ou une connivence avec la tierce personne) dans le tour de carte que vous venez de voir en l'automatisant. ≫

Dans le tour qui vient de vous être présenté, après le mélange des cartes effectué par votre camarade, après le choix des mouvements de cartes effectué sur chaque pile, celui-ci a réussi à faire ressortir les deux même cartes à la fin.

Pour espérer comprendre comment fonctionne ce tour et au moins savoir s'il y a eu une manipulation qui vous est passée inaperçue, on souhaite le traduire en un programme qui effectue les opérations que vous avez vues : on remplace le magicien par un programme qui sera forcément rationnel et déterministe.

En faisant tourner ce programme avec différentes séquences d'actions, on espère comprendre s'il y a des conditions initiales particulières expliquant quand et comment ce tour de magie fonctionne. Si l'on ne trouve rien de concluant, c'est vraisemblablement que votre chargé de TD a fait une passe cachée qui vous a échappée ou que votre camarade était en fait complice.

Q 1.1. Quelle structure de données choisir pour représenter un paquet de cartes ?

Q 1.2. Identifiez les grandes étapes . . .du tour.

Q 1.3. Quelles sont les traitements de la dernière étape ? Combien de fois sont-ils effectués ?

Q 1.4. Comparez les opérations Q1.2-2 et Q1.3-2, que remarquez-vous ?

Q 1.5. De l'algorithme esquissé en Q1.2, de quelles fonctions avez-vous besoin, que font-elles, quels sont leurs domaines ?

Q 1.6. Quel est l'algorithme de la fonction `cut` issue de la question Q1.5 ?

Q 1.7. Quel est l'algorithme de la fonction `split_deck_in_2` issue de la question Q1.5 ?

Q 1.8. Quelle est la forme de la fonction `remove_first` identifiée en Q1.3, qui permet de supprimer la carte se trouvant au-dessus d'un paquet ? Quels sont ses domaines d'entrée et de sortie ?

Q 1.9. Donnez la forme de la fonction `play` qui prend en argument les 2 paquets de cartes et exécute les mouvements dans chaque sous-paquet jusqu'à l'obtention des deux dernières cartes pour vérification d'égalité.

Q 1.10. Il ne vous reste plus décrire la forme de la fonction ≪principale ≫ qui orchestre le tour selon l'analyse que vous avez faite en Q1.2.

2 Implémentation

Q 2.1. Implémentez en `C` toutes les fonctions identifiées et dont vous avez esquissé les algorithmes.

3 Conclusion

Q 3.1. Alors, conclusion ?

S'il vous reste du temps ou pour continuer après la séance.

4 Stop aux allocations dynamiques

Dans l'exercice précédent, chaque manipulation sur les piles de cartes donnait naissance à un ou des nouveaux tableaux. Nous avons donc fait de nombreuses allocations dynamiques. Pour autant, on peut se convaincre aisément que toutes ces manipulations auraient pu être faites au sein du tableau initial.

Le but de cet exercice est de reprendre le programme précédent et de ne plus créer de nouveaux (sous)-tableaux.

Q 4.1. Comment peut-on réécrire la fonction `cut` afin qu'elle effectue ses modifications directement dans le tableau reçu en argument ? Que deviennent ses domaines d'entrée et de sortie ?

Q 4.2. C'est bien entendu maintenant au tour de `split_deck_in_2` de subir le même sort. Quels sont ses domaines d'entrée et de sortie désormais ? Puis quel est son nouvel algorithme ?

Q 4.3. Que devient la fonction `remove_first` de la question 1.8 ?

Q 4.4. Que devient la fonction `play` de la question 1.9 ?

Q 4.5. Que devient la fonction `play` de la question 1.9 ?

Il est même possible de supprimer l'allocation dynamique du tableau initial `full_deck`. En effet, vu que dans cet exercice il est de taille fixe, on peut l'allouer statiquement. On remplace le `malloc` par :

int full_deck [DECK_SIZE] ;

et l'on supprime le

free (full_deck) ;

final.

Le code complet de la solution n'est pas inclu dans ce PDF mais il peut être consulté dans l'archive de correction qui vous est fournie (fichier `magic-opt.c`).