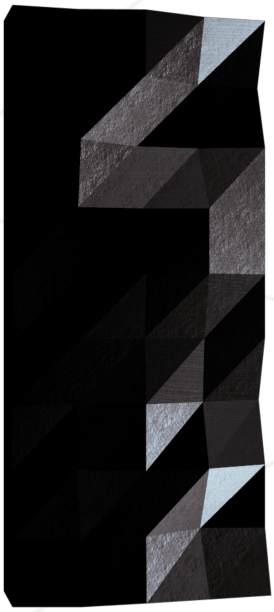

Rush du module Algorithmie I

Puissance4

42 staff staff@42.fr

Résumé: Ce rush est là pour vous faire réfléchir sur des algorithmes et des techniques de programmation efficaces. Ou au pire, pour vous faire réfléchir tout court.

Table des matières

Chapitre I

Préambule

Joshua : Greetings, Professor Falken.

Stephen Falken : Hello, Joshua.

Joshua : A strange game. The only winning move is not to play. How about a nice game of chess ?

Chapitre II

Sujet

II.1 Partie obligatoire

Si vous avez eu une enfance heureuse, vous avez probablement joué au célèbre jeu de société `Puissance 4`. Si ce n'est pas le cas, eh bien c'est le moment de rattraper ce manque.

Comme vous l'avez compris en lisant le titre de ce document, le but de ce rush est d'écrire un jeu de `Puissance 4`. C'est déja un sujet génial en l'état, mais sachez que vous allez également devoir écrire une IA contre laquelle jouer. C'est merveilleux non ?

- Vous devez implémenter les règles disponibles içi.
- A la différence des règles de la version officielle du jeu, votre grille de jeu ne fera pas systématiquement 6 lignes et 7 colonnes. Le nombre de lignes et de colonnes sera transmis à votre programme sous la forme de deux paramètres numériques. Toutefois, la taille minimum acceptable de la grille est la taille de la grille officielle. Dans le cas où les deux paramètres ne respectent pas ces conditions ou n'ont aucun sens, votre programme doit le signaler et quitter proprement.
- Le joueur humain et l'IA jouent à tour de rôle. Le premier à jouer est determiné aléatoirement au début de la partie.
- Au début de la partie, à la fin de la partie et entre chaque coup, votre programme doit afficher dans le terminal l'état de la grille dans un format clair pour des tailles de grille raisonnables. On tolèrera un affichage moins clair pour des grilles de grande taille. Par affichage clair, on entend que les pions de chacun des deux joueurs ainsi que leurs positions relatives et absolues sont simples à identifier.
- A chaque tour du joueur humain, votre programme doit l'inviter à saisir la colonne sur laquelle le joueur souhaite jouer. Si la donnée saisie n'est pas valide, votre programme doit inviter le joueur humain à saisir une nouvelle valeur jusqu'à ce que celle-çi soit valide.
- Contrairement à `Joshua` dans le film `Wargames`, votre IA n'a pas le droit de ne pas jouer. De plus, votre IA doit tenter de gagner, il est hors sujet d'écrire une IA qui joue au hasard. Pour terminer, un IA qui prend plusieurs secondes pour jouer n'est pas une bonne IA, la votre doit répondre la plus vite possible.

- A la fin de la partie, votre programme doit indiquer clairement quel joueur a gagné ou bien si la partie est nulle puis quitter proprement.

II.2 Partie Bonus

Pour vous permettre d'aller plus loin, nous vous proposons une partie bonus à ce rush. Pour que les points de la partie bonus soient comptabilisés, vous devez obtenir au moins 18/20 à la partie obligatoire, quelque soit la qualité de votre partie bonus.

Voici quelques idées de bonus potentiellement intéressants :

- Une interface graphique. Pour cela, vous pouvez utiliser la bibliothèque graphique que vous souhaitez (y compris `termcaps` et `ncurses`). Vous devez ajouter un paramètre à votre programme pour activer ou non votre interface graphique. Lorsque l'interface graphique est désactivée, votre programme doit se comporter exactement comme la partie obligatoire le décrit. Lorsque l'interface graphique est activée, vous devez respecter la partie obligatoire concernant l'affichage dans le terminal, mais les entrées pourront se faire directement depuis votre interface graphique. Faites très attention à ce que votre programme puisse compiler et linker normalement sur l'ordinateur de votre correcteur lors de la soutenance. Installer 3Go de bibliothèques lors de la soutenance ne lui plaira probablement pas. De plus, il est interdit de rendre la bibliothèque graphique ou ses sources sur votre dépot `GIT`, cela sera éliminatoire. Soyez intelligents et élégants dans votre solution à cette problématique.
- Une IA évoluée utilisant un algorithme précis et correctement implémenté dont il sera possible de règler la difficulté au lancement du programme ou au début de la partie. Bien entendu, les consignes de la partie obligatoire concernant l'IA doivent être respectées. Vous devrez prouver à votre correcteur que votre IA respecte à la fois les consignes de la partie obligatoire et de la partie bonus, ainsi que lui indiquer l'algorithme que vous avez utilisé, pourquoi celui-çi est adapté et lui prouver que votre implémentation de cet algorithme est correct pour mériter vos points.

Chapitre III

Rendu

- Vous devez rendre à la racine de votre dépôt de rendu un fichier `auteur` contenant vos 2 logins, chacun sur une ligne, et suivis d'un '\n' :

```
$>cat -e auteur
xlogin$
ylogin$
$>
```

- Votre programme devra s'appeler `puissance4`.
- Vous devez avoir un `Makefile` avec les règles usuelles. Dans le doute, mettez toutes les règles connues.
- Seul le contenu présent sur votre dépot sera évalué en soutenance.
- Dans le cadre du bonus de l'interface graphique, il est interdit de rendre les sources ou une version binaire de votre bibliothèque graphique sur votre dépot de rendu. Soyez intelligents et élégants dans votre solution à cette problématique.

Chapitre IV

Fonctions autorisées

- `read(2)`
- `write(2)`
- `malloc(3)`
- `free(3)`
- `errno`
- `strerror(3)`
- `rand(3)`
- `srand(3)`
- `time(3)` (pour la seed de srand)

Chapitre V

Consignes

Les consignes suivantes feront toutes parti du barème de soutenance. Soyez très attentifs et soigneux dans leur application car elles sont sanctionnées par un 0 sans appel.

- Au cas ou quelqu'un en douterait, ce rush est bien entendu à écrire en `C`.
- Ce projet ne sera corrigé que par des humains.
- Votre projet doit être à la Norme. Une erreur de Norme est éliminatoire, y compris dans votre `libft`.
- Vous devez gérer les erreurs de façon sensible. En aucun cas votre programme ne doit quitter de façon inattendue (Segmentation fault, bus error, double free, affichage de non sens, etc) ou partir en boucle infinie, que ce soit dans la partie obligatoire ou dans la partie bonus. Une telle erreur est éliminatoire. Non, vous n'avez pas besoin de `Valgrind` pour cela.
- Toute mémoire allouée sur le tas doit être libérée proprement. Oublier de désallouer de la mémoire est éliminatoire.
- La valeur de retour de tous vos appels système doit être vérifiée. Un oubli est éliminatoire. On tolérera de ne pas verifier la valeur de retour de `write(2)`. On définit par "appel système" toute fonction dont le man se trouve dans la section 2 sur les Macs de l'école.
- Vous pouvez utiliser votre bibliothèque `libft`. Pour cela, vous devez en rendre les sources à la racine de votre dépot dans un dossier nommé `libft`. Votre bibliothèque devra être compilée en même temps que votre rendu et linkée avec celui-çi. Votre `libft` doit être à la Norme. Utiliser votre `libft` pour contourner la Norme est éliminatoire.

Bon courage à tous pour ce rush !