Partiel C# - Sujet 1
Vendredi 10 Janvier 2014 9h - 12h

1 Consignes de rendu

A la fin de ce TP, vous serez dans l'obligation de rendre une archive respectant l'architecture suivante :

```
-- rendu-tpcs1-login_x.zip
   |-- login_x/
       |-- AUTHORS
       |-- README
       |-- Bonus.txt
       |-- Ex1/
           |-- Ex1/
           |-- Ex1.sln
       |-- Ex2/
           |-- Ex2/
           |-- Ex2.sln
       |-- Ex3/
           |-- Ex3/
           |-- Ex3.sln
```

Les différentes fonctions demandées dans cet examen devront être codées dans un fichier .cs nommé suivant le format suivant : Ex[1-2-3].cs
Bien entendu, vous devez remplacer `login_x` par votre propre login.

Fichier AUTHORS

Le fichier nommé `AUTHORS` doit se trouver à la racine de votre rendu. Il devra contenir tous les auteurs du projet (en l'occurrence, VOUS seulement). Le format, pour chaque auteur, est le suivant :
- Le caractère '*' suivi d'un espace.
- Le login de l'étudiant.
- Un retour à la ligne.

Conseils et remarques

Lisez le sujet dans son intégralité avant de commencer. Une fois que vous l'avez lu, relisez-le.
Réalisez les exercices dans l'ordre car ils sont en difficulté croissante.
N'oubliez pas de vérifier les points suivants avant de rendre :

- Le fichier `AUTHORS` doit être au format habituel (rappelé ci-dessus).
- Aucun dossier `bin` ou `obj` dans le rendu.
- Les fonctions demandées doivent respecter le prototype donné
- **Le code doit COMPILER**.

2 Consignes

1. Pour chaque exercice, dans son dossier de rendu, vous créerez un nouveau projet en console avec Visual Studio, que vous nommerez selon le nom de l'exercice en question.
2. Vous indenterez votre code de manière à ce que cela soit lisible pour nous, correcteurs. Pour cela, souvenez-vous du raccourci Ctrl-K Ctrl-F.
3. Sauf si explicitement indiqué dans le sujet, les seules fonctions autorisées sont :
   (a) Console.Write, Console.WriteLine, Console.Read, Console.ReadLine
   (b) Array.GetLength, String.Length

3 Exercice 1 : Préliminaires (Ex1.cs)

3.1 De la chaîne de caractères à l'entier

Dans cet exercice, vous devez convertir une chaîne de caractères contenant des chiffres en un entier (exemple : "2016" –> 2016).
Les cas d'erreurs (si la string s ne représente pas un entier, etc...) ne seront pas testés.

Prototype :

```
public static int MyAtoi(string s);
```

3.2 Fonction de hachage

Dans cette partie de l'examen, vous devez réaliser une fonction de hachage.

> "On nomme fonction de hachage une fonction particulière qui, à partir d'une donnée fournie en entrée, calcule une empreinte servant à identifier rapidement, bien qu'incomplètement, la donnée initiale. Les fonctions de hachage sont utilisées en informatique et en cryptographie."

Notre fonction prend en argument une chaîne de caractères et renvoie un entier compris entre 0 et 99. Le résultat est le modulo 100 de la somme des valeurs ASCII de chaque caractère de la chaïne.

Protytype :

```
public static Hash(string s);
```

Exemples :

```
hash("a") // La valeur ASCII de 'a' étant 97, 97 modulo 100 vaut 97.
97
hash("abc")
94
hash("It is not that hard")
22
```

4 Exercice 2 : Niveau intermédiaire (Ex2.cs)

Dans cette partie de l'épreuve, vous appliquerez ce que vous avez appris du passage par référence, et ferez appel à votre intuition.

4.1 Echange de valeurs

Dans cet exercice, vous devez réaliser une fonction qui échange les valeurs de deux variables de manière à ce qu'en sortant de cette fonction les deux variables aient effectivement changé de valeur.

Prototype :

```
public static Swap(ref int a, ref int b);
```

4.2 Initialisation d'un tableau

Dans cet exercice, vous devez réaliser une fonction qui initialise un tableau d'entier de taille n avec la valeur -1 et qui retourne ce tableau.

Prototype :

```
public static long[] InitArray(long n);
```

5 Exercice 3 : Puissance 4 (Ex3.cs)

Voici maintenant l'exercice majeur de cet examen. Le titre parle de lui même, vous allez devoir coder un puissance 4 en console. Cet exercice se décomposera en plusieurs parties, et nous essayerons de vous guider au fur et à mesure. Mais n'oubliez pas que comprendre son code est déjà une voie vers la réussite.

Les seules fonctions autorisées durant cet exercice sont celles précisées en début de sujet. En cas d'hésitation, les assistants présents dans la salle seront ravis de vous aider (aider ne signifiant pas "faire le travail à votre place").
Les prototypes de vos fonctions devront respecter scrupuleusement ceux donnés dans ce sujet. Il n'est cependant pas interdit de décomposer votre code en sous fonctions.

5.1 Introduction et présentation

Vous jouerez sur un plateau de dimension (i,j) de votre choix, tant que ces valeurs permettent à un des deux joueurs de gagner. Sachez que lors de la correction, vos fonctions seront appelées avec des valeurs différentes. Par conséquent, vous êtes libre de tester votre code avec n'importe quelle valeur.
Voici la disposition de notre puissance 4 :

Deux joueurs s'affrontent sur un plateau. Le joueur 1 disposent des jetons 'X', et le joueur 2 des jetons 'O'. Le premier joueur arrivant à aligner 4 jetons dans n'importe quelle direction l'emporte.
Comme vous l'aurez deviné, les jetons étant les caractères 'X' et 'O', le plateau sera représenté sous forme de tableau de caractères de dimension 2. Vous devrez implémenter le gameplay, ainsi que l'affichage du plateau. Les indices des tableaux commencent à 0. La notation (x, y) représente la case sur la ligne x à la colonne y.

5.2 Initialisation du tableau

Commençons doucement (tout doucement). Vous devez ici initialiser le tableau "board". Dans notre cas, le tableau de départ, ainsi que tout les cases non occupées lors du reste de la partie seront considérés comme des espaces (caractère ' ').

Prototype :

```
private static void InitGameBoard(char[,] board);
```

5.3 Get & Set

Voici les deux fonctions fondamentales qui vous permettrons de comprendre le fonctionnement d'un tableau si ce n'est pas déjà fait, et de le manipuler lors des différentes actions de jeu.

Tout d'abord, GetCase :
Prototype :

```
private static char GetCase(int c, int r, char[,] board);
```

Ici, c et r désignent respectivement la colonne et la ligne auxquelles nous voulons accéder. Vous pouvez considérer que r et c sont des paramètres valides, respectant les limites du tableau.

Voici venue la plus importante, SetCase :
Prototype :

```
private static bool SetCase(int c, int r, char token, char[,] board);
```

Cette fonction place le caractère "token" dans le plateau de jeu "board", sur la ligne r et à la colonne c. Elle devra renvoyer les valeurs suivantes :

- False si les paramètres donnés (c, r) sont invalides vis à vis du tableau, ou si le plateau de jeu contient déjà un jeton à cet endroit.
- True si l'insertion s'est passée sans problème.

Veillez à ce que cette fonction ait le comportement voulu. Elle sera une des pièces maitresses de votre puissance 4.

5.4 Dessin

Mettre des jetons dans des cases, c'est bien, mais on s'ennuie rapidement lors d'un partiel. Voici donc une fonction qui devrait réveiller vos talents d'artistes :
Prototype :

```
private static void DrawGameBoard(char[,] board);
```

C'est cette fonction qui vous permettra (et permettra aux assistants) de comprendre ce qui se passe dans votre code. Attention, chaque caractère compte, et vous devrez respecter scrupuleusement le format suivant :

```
  1   2   3   4   5   6
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   | X |   |   | O |
| O |   | O | X |   | X |
-------------------------
```

Chaque case devra être représentée selon la suite de caractères suivante : '|', un espace, le caractère de la case, un espace, '|'. AUCUNE ligne ne se finit par un espace. Ainsi, 5 est le dernier caractère de la première ligne.

Pour rappel, les indices commencent à 0. Le coin haut-gauche représente ainsi la position (0,0), et le coin bas-droit la position (5,5).

5.5 Insertion et Mécanismes de jeu

5.5.1 Insertion à la descente

Vous voici avec des jetons dans des cases et un magnifique plateau en ASCII, mais d'aussi loin que portent vos souvenirs, le jeu consistait à laisser tomber des jetons dans des cases, et non pas à démonter le plateau à chaque tour de jeu ? Vous l'aurez voulu, voici les fonctions qui vous permettront de faire ça "virtuellement". (PS : Le réglement du bocal interdit de démonter la RAM pour y placer vos jetons).

Selon toute logique, lorsque qu'on laisse tomber un jeton, il s'arrête un cran au desssus du dernier jeton ajouté dans cette colonne. Voici la fonction qui calculera ceci :
Prototype :

```
private static int GetPosition(int c, char[,] board);
```

Cette fonction qui prend en paramètres la colonne à laquelle on désire ajouter un jeton, ainsi que le tableau en lui même, devra retourner les valeurs suivantes :

- (-1) si La colonne est pleine, ou si le numéro de colonne est invalide.
- La position d'ajout dans la colonne si l'insertion a réussi.

En reprenant l'exemple du schéma ci-dessus, avec le tableau tel qu'il est représenté (dimensions : 6x6), un appel à la fonction GetPosition(0, board) retournera 4.
Un appel de GetPosition(2, board) retournerait 5.
Enfin GetPosition(3, board) retournera 3.

5.5.2 Time to play

Allez on enchaîne ! Voici la fonction de jeu PlayToken, qui servira comme interface pour placer un jeton dans une des colonnes.

```
private static bool PlayToken(int c, char token, char[,] board);
```

Cette fonction devra tenter de placer le jeton "token" dans la colonne "c".
Si tout s'est bien passé, PlayToken devra renvoyer True.
En cas d'erreur (colonne remplie ou indice invalide), elle devra renvoyer False.

En cas d'erreur, vous devrez afficher sur la console les messages suivants :
- "Full column, choose a new one!" si la colonne est pleine.
- "Invalid column number, choose a new one!" Si l'indice c est invalide.

5.6 And the winner is...

Allez, c'est bientôt fini ! Vous avez dessiné des tableaux, placé des jetons, et vous allez maintenant connaître le gagnant. Le prototype de la fonction parle de lui même :

```
private static char WinnerIs(char[,] board);
```

Voici une des fonctions les plus longues de cette partie, en terme de code et non de temps. N'hésitez pas à la découper en plusieurs sous fonctions. Votre code vous paraîtra plus clair, et vous aurez ainsi beaucoup moins de mal à débugger vos fonctions.

Pour rappel, on considère qu'un joueur gagne si quatre de ses jetons se retrouvent alignés dans un tableau, de manière horizontale, verticale, ou diagonale.
Cette fonction retourne le caractère correspondant au gagnant si il y en a un ('X' ==> Joueur 1, 'O' ==> Joueur 2). Si personne n'a encore gagné, WinnerIs retourne le caractère espace ' '.
Exemples :

```
/* Joueur 1 gagne */
  1   2   3   4   5   6
|   |   |   |   |   |   |
|   |   |   |   |   |   |
| X | X |   |   |   |   |
| O | X | O |   |   |   |
| O | X | X |   |   | O |
| O | O | O | X |   | X |
-------------------------
```

```
/* Joueur 2 gagne */
  1   2   3   4   5   6
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   | O |
|   |   |   |   | O | X |
|   |   | X | O | X | O |
| O |   | O | X | X | X |
-------------------------
```

5.7 Le Jeu

Voici venu le moment de rassembler votre code, et de jouer contre vos amis (après l'examen). Une partie de la boucle de jeu sera contenue dans la fonction suivante :

```
private static void Play(char[,] board);
```

Cette fonction devra suivre la procédure suivante :

1. Afficher "Player X, please choose a column : " ou "Player O, please choose a column : ", selon que ce soit au joueur 1 ou au joueur 2 de jouer. N'oubliez pas l'espace après les deux points. Vous ne devez pas revenir à la ligne.

2. Demander à l'utilisateur de rentrer un numéro de colonne sur l'entrée standard. Pour cette étape de l'examen et seulement cette étape, vous avez le droit à la fonction "Convert.ToInt32()". Si l'entrée n'est pas un nombre, recommencer l'étape.
3. Utilisez les fonctions précedentes pour placez le jeton du joueur. Si le mouvement en question n'est pas valide, retourner à l'étape 2.

Exemple :

```
Player X, please choose a column : no
Player X, please choose a column : 2
```

5.8 La boucle est bouclée

```
public static void Connect4(uint nbrow, uint nbcolumn);
```

Vous voilà à la fin. La concrétisation de ce partiel. Voici la procédure finale qui devra être implémentée dans cette fonction :

1. Création du tableau et initialisation de celui-ci.(Dimensions nbrow X nbcolumn)
2. Dessiner la grille initiale
3. Afficher une ligne vide
4. Demander au joueur courant de jouer
5. Afficher une ligne vide
6. Dessiner la grille
7. Si un gagnant est détecté, afficher "Player X wins !" ou "Player O wins !", sinon retournez à l'étape 3. (Il y a bien un espace entre "wins" et "!")

Nous ne vous demandons pas de gérer les cas d'égalité. Voici des exemples qui illustrent ce que votre affichage doit rendre :

```
  1   2   3   4   5   6
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   | O | X |
|   |   | X | O | X | O |
| O |   | O | X | X | X |
-------------------------

Player O, please choose a column : 5

  1   2   3   4   5   6
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   | O |
|   |   |   |   | O | X |
|   |   | X | O | X | O |
| O |   | O | X | X | X |
-------------------------
Player O wins !
```

```
  1   2   3   4   5   6
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
-------------------------

Player X, please choose a column : 95
Invalid column number, choose a new one !
Player X, please choose a column : 2

  1   2   3   4   5   6
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   | X |   |   |   |   |
-------------------------
```

5.9 Bonus

Vous pouvez implémenter tous les bonus que vous voulez, mais ceux ci doivent apparaître dans une classe appelée Bonus.cs, et vous devrez en faire part explicitement dans le Bonus.txt.
Les fonctions faisant partie du rendu obligatoire ne doivent en aucun cas être modifiées !
Vous êtes libres sur cette partie, néanmoins, voilà des classiques :

Gestion de l'égalité, sons, retour en arrière (selon un appui sur une touche par exemple), intelligence artificielle, affichage du tableau amélioré, etc...
N'OUBLIEZ PAS de supprimer vos répértoires obj/ et bin/, et de suivre la structure du rendu à la lettre.