PARTIEL C# - Sujet 2

Friday 10 January 2014, 12h30-15h30

1 Report's consign

At the end of this partial exam , you'll have to provide a file which respects the following architecture :

```
report-partiel1-login_x.zip
|-- report-partiel1-login_x/
    |-- AUTHORS
    |-- SkillCheck
        |-- SkillCheck.cs
    |-- shogi.sln
    |-- shogi
        |-- Everything but bin/ et obj/
```

Don't you forget to check the following points before you give back :

AUTHORS file

The `AUTHORS` file must be at the root of your report. It must contain every Authors of the project (In this case, only you). The format for every Authors is the following :

- Character '*' followed by a space.
- Student's login.
- A newlin.

Advices

Read the whole subject before you start. Once you read it, read it again. Try to make the exercices in the order, as the difficulty is growing.
Don't you forget to check the following points before you give back :

- The `AUTHORS` file must be on usual size.
- No folders `bin` or `obj` in the report.
- Function must follow the given prototype.
- **Your code must compile**.

Before you realize any exercise implementation do read what follows to understand you are asked. You can write as many functions and declare as many variables as you wish. Good luck

2 preliminary

For this part you must give back the exercices in a file SkillCheck.cs.

2.1 It's just the beginning (Approximately 5 minutes)

Write the function `hello` which ask for your name and repeat it with hello.

Prototype :

```
static void hello();
```

2.2 Keep going on (Approximately 15 minutes)

Write the function `pow` which computes x pow n, where n is a natural integer.

Prototype :

```
static int pow(int x, int n);
```

Test your might :
Optimise your code and handle the negative pow.
Tips : $2^4 = 2^2 \times 2^2$

2.3 Finally (Approximately 20 minutes)

Write the function `fibonacci` which computes fibonaccu at rank n in iterative way. Recursive function are banned for this exercice.
Remind :

- fibonacci(n) = fibonacci(n - 1) + fibonacci(n - 2)
- fibonacci(0) = 0
- fibonacci(1) = 1

Recursive function are banned for this exercice !

Prototype :

```
static int fibonacci(int n);
```

3 Introduction

3.1 Goals

During this partial exam we'll use tracks studied in previous TP

- Use of console in C#
- Loops and conditions
- Classes (Bonus)
- Boards
- The main function

4 Subject

4.1 The Hasami Shogi

During this partial exam you'll have to code a modified version of Hasami Shogi , simplified version of Shogi , Japanese chess game.

Hasami Shogi game is traditionally played in Japan . Its characteristic is to be played according two variants, one being a position game the other a confrontation game. According to the variant you play the target is either to build a string of five pieces (or pawns) or to capture the pieces of the opponent. Captures are carried out by sandwiching , pieces move like the tower in chess game ou by short jump. Then it appears compulsory to set traps to opponent pieces. [1] This is one of the most popular variant of Shogi in Japan, the Japanese chess game. The word hasami means sandwiching which shortly describes the mean to capture a piece In this partial you'll be asked to implement the confrontation variant

4.2 The game

The game begins with a game board of 10*10 where each player owns a set of pieces of his color (white or black). [2]. At each round the player chooses a piece of his color and moves it vertically or horizontally of as many cases as he wishes, however he can't jump over the other pieces. [3] To capture an opponent piece you just have to sandwich it. In the traditional game you can capture a set of pieces, this won't be asked to you but could be a pleasant bonus as much as capture in corners The game is over when a player has captured all the opponent pieces but one, then the opponent becomes unable to capture any piece.

4.3 TO BE REMINDED

4.3.1 Use of console in C#

The use of the console in C# is carried out through console class Here are a few functions that may be of any use :

1. Console.Write
2. Console.WriteLine
3. Console.Read
4. Console.ReadLine

4.3.2 Two dimensions boards

Your game board will be represented as a two dimensions board : here is an example

```
int m = 10;
int n = 5;
int[][] tab = new int[m][];
for (int i = 0; i < m; i++)
    tab[i] = new int[n];
```

1. "It's a trap" Admiral Ackbar, a long time ago
2. Traditionally, 9*9 and 18 pieces for each player.
3. In traditionnal game, a jump is possible if the piece start next to the jumped piece.

5 Exercices

5.1 step 0 : create a board (Approximately 10 minutes)

Write the function `init_board` which creates a two dimensions board initialized with spaces. This function must also initialize player pieces on that boards.

Prototype :

```
static char[][] init_board();
```

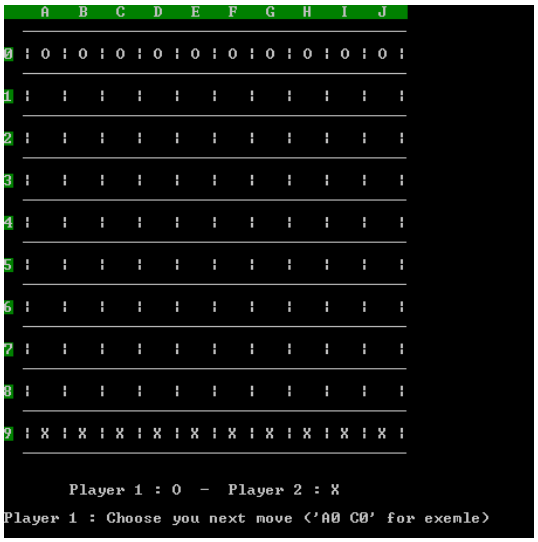
5.2 Step 1 : display board (Approximately 20 minutes)

Write the function `print_board` which prints the board on the console.

Prototype :

```
static void print_board(char[][] board);
```

Example :



Test your might : Add line and rows number, as in the exemple image.

5.3 Step 2 : Try again ! (Approximately 30 minutes)

Write the function `move_is_valid` which makes sure that the player's move is valid. A valid move is an horizontal or vertical move that didn't cross the others pieces.

Prototype :

```
static bool move_is_valid(char[][] board, int x1, int y1, int x2, int y2);
```

Test your might :
Handle the jumps like in the original game. A piece can jump only if it starts close to the jumped piece and if the case behind is free. The piece goes to this case.

5.4 Step 3 : Move (Approximately estimée 5 minutes)

Write the function `move_pawn` which moves a piece on the board. If the move is valid then the piece go to the new location and the older position is deleted.

Prototype :

```
static void move_pawn(char[][] board, int x1, int y1, int x2, int y2);
```

Test your might :
Handle the jump move.

5.5 Step 4 : Eat me ! (Approximately estimée 15 minutes)

Write the function `eat_pawn` which covers the board and captures a piece if it is sandwiched by to opponent pieces . The piece is deleted only if it is sandwiched horizontally or vertically, we don't care about diagonal.

Prototype :

```
static void eat_pawn(char[][] board);
```

Test your might :
Add multiple pieces capture (to surround or encircle a set of pieces deletes the whole set), and the capture of pieces in the corners of the board (a piece is captured in a corner if it gets blocked by two opponent pieces)

5.6 Step 5 : Mission completed (Approximately 20 minutes)

Write the function `game_over` which returns true if a player wins. Remind : A player has lost when he only has one piece left on the board.

Prototype :

```
static bool game_over(char[][] board);
```

Test your might :
Add a set of five out of the basic zone as a condition to victory (vertical horizontal or diagonal serie).

5.7 Step 6 : Game loop (Approximately 40 minutes)

Write the function `game_loop` which :
- Initialize the board
- Handle player's turn
- Get back the users entries
- Check if the move is valid
- Move a piece
- Eat captured pieces.
- Check if someone won.

Prototype :

```
static void game_loop();
```

Test your might :
Add a function newgame to start a new game after playing one.

5.8 Bonus : Make a class

To get this bonus you'll have to declare a class `Game` and enter your code inside . This class will take your board as an attribute and this will enable you to delete the board argument from your functions. You'll have to add the function `Game` hat creates a new object of the class `Game` in order to begin a new game.

It's dangerous to code alone !