# Bulls & Cows

## Overview
This assignment requires you to develop a simple game called Bulls & Cows. There are nine compulsory tasks in this assignment to complete.

Before starting the assignment, read through and gain an understanding of the requirements.

## Criteria
Your assignment will be marked based on the correctness of your program and your programming style. Here are some questions to consider for the programming style:

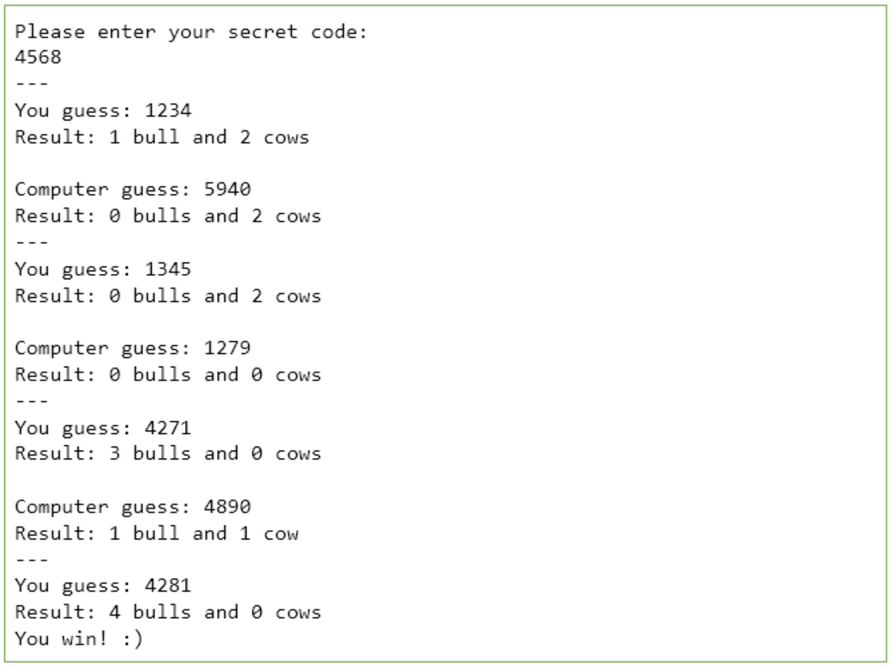
* Is the code well-structured?
* Is the code self-explanatory?
* Can you understand the code easily?
* Are all variables properly defined with meaningful and clear code?
* Is there any commented-out code?

Marks will be deducted if your code cannot be compiled or has a bad programming style.

## Assignment Details
The bulls and cows game is a code-breaking game designed for two or more players. Each player chooses a secret code of 4 digits from 0 – 9. The digits must be all different. The goal of the game is for each player to guess the other player's secret code.

The players in turn present their guesses to the opponents. The opponents respond by telling the players:
1. The number of bulls, i.e. the number of matching digits in their right positions, and
2. The number of cows, i.e. the number of matching digits but in different positions.

For example, if the computer's secret code is 4281, the match responses for the following guesses are shown as follows:


```
Please enter your secret code:
4568
---
You guess: 1234
Result: 1 bull and 2 cows

Computer guess: 5940
Result: 0 bulls and 2 cows
---
You guess: 1345
Result: 0 bulls and 2 cows

Computer guess: 1279
Result: 0 bulls and 0 cows
---
You guess: 4271
Result: 3 bulls and 0 cows

Computer guess: 4890
Result: 1 bull and 1 cow
---
You guess: 4281
Result: 4 bulls and 0 cows
You win! :)
```

For more information about the game itself can be found [here](https://en.wikipedia.org/wiki/Bulls_and_Cows).

### Requirements
The main goal of this assignment is to develop the bulls and cows game that allows a single player to play interactively against the computer. The game stores two secret codes, one from the player and one from the computer. The player and the computer will try to guess each other's secret code. Both the player and the computer only have seven attempts for guessing the secret code. If the player enters an invalid input, the game should ask the player to try again. The game also lets the player choose the difficulty level to play against the computer.

There are three levels: easy, medium, and hard. The details of the difficulty levels will be described in later sections.

In addition, the game can read from a text file that contains multiple guesses from the player, and save the results to another text file. You don't have to do this now, you will do this after the discussion on IO and File IO.

For this module assignment, you'll complete a series of tasks as you work your way toward a fully functional implementation. As you complete each task, it might be a good idea to save a working version of the program at that point.

## Task One: Design and Feedback (5 pts)
For this assignment, there is very little which is already given to you. Through this assignment, you'll gain experience in designing and building a complex program from scratch. Before

starting to code, don't forget to design your classes and methods (i.e. create UML class and sequence diagrams)! You should apply the concepts you have learned so far in the course. You should not have everything in one class, and try to promote code reuse as much as possible. You must have at least one use of inheritance in this assignment.

Using either pen & paper or any diagramming tool of your choice, prepare a UML class diagram that shows all the classes and important methods of your Bulls & Cows implementation, along with the appropriate relationships showing how these classes fit together. Save this class diagram (as a PNG or JPEG) within the module assignment folder. Save it as BullsAndCowsDiagramTask1_YOURLASTNAME.[png/jpg] file. Use what you've learned in OOP and apply the OOP Concepts on your design.

Note: It is OK if your implementation doesn't match your initial design 100% - things do change! You will document this process in the later task. You will include all the feedback received as part of your reflection.

## Task Two: The Beginning (10 pts)
Implement the first part of the game allowing the player to guess the computer's secret code. The computer randomly generates the secret code at the beginning of the game, which it then lets the player guess. Remember that when generating the computer's secret code, each of the four digits must be different. Note that the player only has seven attempts to guess the secret code. The prompt for player input results for each guess and the final outcome (i.e. whether the player has won the game or not) should be displayed appropriately on the console.

## Task Three: Easy AI (10 pts)
Save your Task 2 code on a different package.

Modify your code so that the player can now also enter a secret code when the game begins, which the computer must guess. Remember to verify that the player has chosen a valid secret code. The player and computer each take turn guessing the other's code. The game ends when either side successfully guesses the other's code (resulting in a win for that side), or when each side has made seven incorrect guesses (resulting in a draw).

For this task, have the player play against an easy AI. When the AI makes a guess, it will simply generate a random (valid) guess.

## Task Four: Medium AI (10 pts)
Save your Task 3 code on a different package.

Modify your code so that at the beginning of the game (before the player enters their own secret code), they will be asked to select either an easy or medium AI opponent to play against.

If the player chooses to play against an easy AI, the game should proceed in exactly the same manner as in Task Three. However, if a medium AI is selected, the AI should keep track of guesses it has already made. The AI will not make the same guess twice.

## Task Five: Hard AI
Ignore this.

## Task Six: Reading Guesses from a File (10 pts)
Save your Task 4 code on a different package.

Modify your code so that before the game begins, the player is asked whether they wish to enter their guesses manually, or to automatically guess based on pre-supplied guesses in a file.

If the first option is chosen, then the game should progress in the same fashion as in the tasks above. If the second option is chosen, then the following actions should be taken:

Firstly, the player should be asked to enter a filename. If the player enters an invalid filename, they should be re-prompted until they enter the name of a file that actually exists. This file should then be read and interpreted as a text file, where each line contains a separate guess. For example, a sample file input.txt may contain the following text:

```
1234
4321
5830
8437
1489
3271
2530
```

You may assume that each line of the file contains a valid guess.

Once the file has been read, then the game should proceed as normal. However, when the player would be prompted to enter a guess, the next guess in the list of pre-supplied guesses should automatically be chosen instead. If there are no more pre-supplied guesses (for example, if the player needs to enter their fifth guess but the file only contains four guesses), then the player should be prompted as normal.

## Task Seven: Saving to a File (10 pts)
Save your Task 6 code on a different package.

Modify your code so that, when the game ends (win, lose, or draw), the player is asked if they wish to save the results to a text file. If they do, then they'll be prompted to enter a filename. The game should then save the following information to the given file:

* The player and computer’s secret number
* Each guess that was made, in the same order as occurred during the game, along with the result of that guess (i.e. how many bulls & cows it got)
* The identity of the winner (or a message stating that the game was a draw)

The data must be readable when opened in a standard text editor. An example results file is illustrated as follows, though exactly how your file is organized is up to you.

```
Bulls & Cows game result.
Your code: 4568
Computer’s code: 4281
---
Turn 1:
You guessed 1234, scoring 1 bull and 2 cows
Computer guessed 5940, scoring 0 bulls and 2 cows
---
Turn 2:
You guessed 1345, scoring 0 bulls and 2 cows
Computer guessed 1279, scoring 0 bulls and 0 cows
---
Turn 3:
You guessed 4271, scoring 3 bulls and 0 cows
Computer guessed 4890, scoring 1 bull and 1 cow
---
Turn 4:
You guessed 4281, scoring 4 bulls and 0 cows
You win! :)
```

## Task Eight: Advanced Configuration (10 pts)
Save your Task 7 code on a different package.

Modify your code so that two of the following values are configurable for Easy AI:
* The maximum allowed number of turns before the game ends in a draw (default is seven)
* The length of a secret code (default is four)
* The allowed characters in a secret code (default are the digits 0 - 9, but letters should be able to be included also. For example, you could say that the allowed characters for a particular game are the digits 0 - 9, and the letters A - F).

To configure these values, you may either read them in from a file or simply have constant variables somewhere obvious in your code, which the user can edit and then re-compile.

Note: It is important that, when configuring these values, the rest of your code should not break! Everything else should continue working as normal.

Please also write the configuration you have implemented and the instructions on how to configure the values in the reflection report.

## Task Nine: Reflection and Self Evaluation (5 pts)
Now that you’ve completed your tasks, it's time to reflect upon your design and implementation.

To begin, create another UML class diagram modeling your source code for Bulls & Cows (again, include this diagram in your repository). You might find that this is different from the initial design you did in Task One. It’s all right.

In a pdf file named ‘BullsAndCowsDiagramTask9_YOURLASTNAME.pdf’ file, write two to four paragraphs reflecting on your learning as well as your design and implementation for this assignment. Here are some questions that might help you to reflect on your learning:

* What have you learned by doing this assignment?
* How well did you think this assignment went?
* What lectures / labs did you find helpful for this assignment?

Your report must also explain the following:
* Why do you believe your design / implementation is good?
* What makes your design good?
* If you do not think your design is good, explain what and how you could improve.
* If your final implementation is different from the initial design, explain why you have made such changes.
* How did you ensure your program worked as expected?
* In addition, write down the advanced configuration you have implemented in the report.
* On a separate page, create a self-evaluation of your task. List all the tasks you’ve completed and give yourself an honest grade for each task. An honesty grade will be applied.

** Honesty grade is worth 30 pts. It will be given if you’ve honestly assessed your MP5. And it will be deducted if there’s any form of dishonesty committed in your MP5, which includes copying codes, letting an AI or someone create the code for you, or a misrepresentation / wrong assessment of your codes and reflections.

For this task, use LETTER page format, 10 pt. font for the body, use Arial or Verdana for the font style, and 1.5 spacing. Make sure you added your diagram from task 1 and your final diagram and in 1 to 2 paragraphs explain the difference.

Personal Grading:
Write a summary of your own grade:

Task 1: [   ] max 5 pts
Why do you deserve this grade?

Task 2: [   ] max 10 pts
Why do you deserve this grade?

Task 3: [   ] max 10 pts
Why do you deserve this grade?

Task 4: [   ] max 10 pts
Why do you deserve this grade?

Task 5: [   ] ignore this.
Task 6: [   ] max 10 pts
Why do you deserve this grade?

Task 7: [   ] max 10 pts
Why do you deserve this grade?

Task 8: [   ] max 10 pts
Why do you deserve this grade?

Task 9: [   ] max 5 pts
Why do you deserve this grade?