Judicial Commission of NSW: Programming Assessment

Please read the instructions below carefully.

Purpose

This programming assessment is designed for us to determine your understanding of programming fundamentals, assess your approach to problem solving, and to provide us insight into your coding style.

Your own work

Obviously there are solutions to this, and similar problems, which can easily be found on the internet. In a work environment, researching what has already been done and not "reinventing the wheel" is a good thing to do, however with this test we are not trying to see how good you are at internet research, or your copy and paste ability. We want to see how you solve the problem.

By sending your solution to us, you are guaranteeing that it is all **your own work**, you have not researched this problem on the internet, and you have not sought advice on the solution from other people.

Before Starting

You may write your solution in any programming language of your choice.
It should take you approximately one to two hours to complete.

We will be assessing your solution across the following three criteria:

Quality	The quality / elegance / clarity of the code and documentation.
Outcome	The correctness of the output.
Error Handling	Your application should handle any and all errors that you feel are sensible to check.

Your Solution

Please place all required files along with a *README.txt* file that explains your solution, and instructions to run it, in a directory named after yourself (excluding spaces) and zipped up.

For example, 'Jane Smith' would create a new directory called *JaneSmith* and put her solution into this directory with a *README.txt* file that contains instructions for running her program. She would then zip up the folder and email that zip file to the assessor.

Programming Assessment: CheckerBoard

You work for a board game company and they have asked you to create a program that can draw a checker board or chess board.

The checker board should be output as plain text from your program.

The checker board program needs to take two arguments:

- **Square size (Integer)** - The first argument is how big each checkerboard square is, eg '3' would generate squares that are 3 columns by 3 rows
- **Board size (Integer)** - The second argument is how big the actual checkerboard should be, eg '2' would generate a checkerboard with 2 columns and 2 rows of squares.

Example Output

First argument: 1
Second argument: 4

```
produces a 4 x 4 checkerboard with 1 x 1 squares:
```

```
      +-+-+-+-+
      |#| |#| |
      +-+-+-+-+
      | |#| |#|
      +-+-+-+-+
      |#| |#| |
      +-+-+-+-+
      | |#| |#|
      +-+-+-+-+
```

First argument: 4
Second argument: 3

```
produces a 3 x 3 checkerboard with 4 x 4 squares:
```

```
      +----+----+----+
      |####|    |####|
      |####|    |####|
      |####|    |####|
      |####|    |####|
      +----+----+----+
      |    |####|    |
      |    |####|    |
      |    |####|    |
      |    |####|    |
      +----+----+----+
      |####|    |####|
      |####|    |####|
      |####|    |####|
      |####|    |####|
      +----+----+----+
```