```
CMPT 213 Assignment 3 Marking Guide
for Dr. Fraser's class: Maze Game

Assignment may be done in pairs, or individually.

Total   = 100 marks

Phase 1: Design
Total [20] Marks
[ 5] CRC
        - Reasonable break-out of classes and high-lever responsibilities.
        - List major class collaborators.
[10] UML Class Diagram
        - Clear OOD:
                * Each class responsible for one thing.
                * Reasonably detailed break-out of classes to handle responsibilities.
                * Each class demonstrates correct encapsulation.
                * Consider use of immutable classes where applicable.
                * Respect the command/query separation guideline when appropriate.
        - Correct and meaningful class relationships
        - Some methods and/or fields listed (but should not have all!)
          (Should be at least enough to cover the big tasks for playing the game)
[ 5] OOD Explained
        - Good description of how two non-trivial tasks are completed.
        - Descriptions should highlight the responsibilities of some classes, and
          rely on the relationships between classes as needed.


Phase 2: Implementation [80 marks]
[30] Maze generation
     Likely deduction per defficiencies
         * -5  internal 2x2 square of walls
         * -5  internal 2x2 square of no-walls
         * -10 lack path to all open cells
         * -5  maze lacks loops (it should sometimes have multiple paths to a cell)
[ 5] Maze display & maze cheat code
[ 5] User interaction handles errors (for example, asking user to re-enter move)
[15] Correct movement of the user, and revealing the maze.
[15] Correct movement and operation of the cats
[10] Correct handling of the cheese, winning and losing
     Includes 1-cheese-to-win cheat code

[0] Correctly follow coding style guide.
    (-30 point max deductions)
        * Must have separate packages for game logic and the UI.
        * Very minor violations have no penalty (ex: having "int myCount=0;" (spacing
wrong))
        * Lose a few marks for consistent problems (like always getting the spacing wrong).
        * Larger penalties possible for horrific code (such as not marked!)

        Some specifics to check
        * JavaDoc-style comment on each class (not needed on methods/fields if clear and
well named)
        * Correct indentation, brackets, spacing
        * Good intention reveiling class, method, and variable names.
        * Easy to read code; refactored if it was complex!
        * Limited depth of nested logic (if-if-if-if-for-if-switch-...)


Forward to Dr. Fraser if...
- Material is suspiciously similar to another submission or code posted online.
```