**Assignment 02 – Marking Scheme**

**General Marking Notes**

* The deadline for grading is Monday, September 30th at 12:00pm (noon).
* We have to proctor the Midterm on Monday, September 30th, so please finish by the deadline!
* If you have any questions email the official marking thread (send it to [cs135-markers@cs.uwaterloo.ca](mailto:cs135-markers@cs.uwaterloo.ca)) with the assignment instructor(s) CC’d (Adrian Reetz: [adrian.reetz@uwaterloo.ca](mailto:adrian.reetz@uwaterloo.ca) and Byron Weber Becker: byron.weber.becker@uwaterloo.ca) so that all ISA’s and TAs can hear about clarifications or changes to the marking scheme.
* If you are not familiar with the design recipe, read the Style Guide on the course website.
* Please email [cs135@uwaterloo.ca](mailto:cs135@uwaterloo.ca) when you are finished marking each week. **Include a list of the common errors you encounter**.
* Guidelines from Assignment 01 carry forward.

**General Notes on the Marking Rubric**

* **Do not deduct more than one level for the same error that occurs in multiple places.** (i.e. if a student is missing the function header in their purpose statements in multiple functions, only deduct one rubric level for this. If another purpose statement error is made, then another rubric level is deducted.)
* Student files can get quite long. **There is no need to thoroughly read through every single line of code.** Instead, focus on various sections of their files to get an idea of a student’s overall understanding on the sections labeled **Global**
* Unless otherwise specified, the marking scheme does not apply to bonus questions
* There are sections dedicated to test coverage on the GRADED\_ASSIGNMENT.ss file (i.e. Q2: Test Cases and Q2: Black Highlighting). This is the first time that they’ve been auto marked. Please double check that the rubric levels assigned to these sections matches the coverage test results. If we hear complaints from students that they’re wrong, we might have to forgo the script and make markers manually assign these grades, so please let us know if there are any issues before we release the grades to students ☺.

|  |
| --- |
| **Q1 Correctness: Cond**  Students should have simplified cond statements that contain no nested conds. **Deduct one rubric level from Q1a, Q1b, Q1c, or Q1d’s correctness** if students commit the following error(s) in those questions.  Errors:   * More than one cond statement per function |
| **Q2: Constant Use**  Students should define **at least 5** of the following constants:   * A character limit with value 10 * A character limit bonus with value 25 * A title penalty with value -50 * A marvel movie bonus with value 500 * A DC movie penalty with value -250 * An actors’ bonus with value 50 * The string “The” * The string “Marvel” * The string “DC” |
| **Q2: Helper Function Use**  Students should have **at least two** substantive helper functions. Suggestions:   * Helper function to calculate bonus/penalty for name longer than 10 char and checking for string “The” in title * Helper function to calculate bonus/penalty for Marvel/ DC movies * Helper function to calculate bonus/penalty for famous actors * Helper function to calculate bonus/penalty for explosions |
| **Q2: Purpose Statements**  A purpose statement should be clear and concise. It should contain a reference to every parameter in the function. It’s fine if the parameter names are chosen to make sense in an English sentence. References to constants are acceptable but not required. **Purpose statements are required for helper functions.**  Errors:   * Missing purpose (or missing function block) * Unclear purpose statement (e.g. describes how the function works rather than what it does) * Missing function header at the beginning of the purpose (i.e. (fn-name parameter1 parameter2)) or does not match actual function header * Not referencing all parameters as they are written in the function header * Extremely long purpose statements; about 5-6 lines long |
| **Global: Contract Correctness**  Correct contracts should be exactly as written at the end of this document (not including format). **Contracts are required for helper functions**.    Select the appropriate rubric level based on how many contracts are correct. Any of the following errors make a contract incorrect:   * Missing contract (or missing function block) * Adding untrue requirements (including restricting “output”) * Incorrect type or incorrect number of types listed (other than the exceptions below)   Exceptions:   * An Int with a requirement that it must be a non-negative number is the same as a Nat (**leave a comment,** but do not deduct any marks) * If students specify unneeded, but true, requirements (for example, a requirement that a Nat must be greater than or equal to 0), **leave a comment,** but do not deduct any marks |
| **Q2: Contract Format**  Contracts should be formatted correctly. If no contracts are included at all, award the student a Level 0; otherwise, mark whichever contracts are present.  Errors:   * Missing uppercase letters to begin type names (Num vs. num) * Missing or incorrect function name * Missing colons after the function name * Brackets are surrounding the function name (i.e. (fn-name)) * Parameter names are included (i.e. fn-name param1 param2 …) * Using incorrect type names (Num vs. Number) * Missing -> (however, => is fine) |
| **Q2: Whitespace/Layout**  Solutions should be indented properly and lines shouldn’t be “too long” or “too short”. Also, there should be blank lines separating function blocks. It is acceptable, but not required, for students to separate function blocks using a row of symbols (such as \*).  Errors:   * Excessively long lines * Missing separators between function blocks (separators can be blank lines or rows of symbols * Design recipe components are not in order * Constant/helper function definition comes after its use in a function * Constant/helper function definition interrupts design recipe |
| **Q2: Names**  Constant, parameter, and helper function names should be descriptive but not too long.  Errors:   * Ambiguous names * Inappropriate naming conventions such as:   + missing dashes between words in a name (with the exception of numbers; that is, anything similar to using cs135 instead of cs-135)   + use of uppercase letters (with the exception of names that are proper nouns)   + use of punctuation or underscores   + use of special characters (with the exception of /,?) |
| **Q2: Code Complexity**  Solutions for this assignment are fairly straightforward. Slight complexities in code are acceptable. Incorrect code should still be marked for Code Complexity.    Errors:   * Including a cond in the answer part of an else as below:   (cond …  [else (cond …)])   * Using eq?, eqv? or equal? Instead of more specific equality predicates (such as = or symbol=?) * Using (boolean=? true x), (boolean=? false x),   (boolean=? x true), or (boolean=? x false)   * Unnecessary helper functions |
| **Q3b: Examples**  Examples should include 2 - 3 **distinct**(check-expect … …) function calls testing the basic functionality of the code. Students may use the same examples as described in the assignment. |

**Contracts**

Q2:

;; box-office-profits: Str Str Nat Nat -> Int

Q3a:

;; intentional-grounding?: Bool Bool Bool -> Bool

Q3b:

;; intentional-grounding-correct?: Bool Bool Bool Bool -> Bool

Q3c:

;; intentional-grounding-penalty: Bool Bool Bool Bool Bool -> Sym