**Assignment 05 – Marking Scheme**

**General Marking Notes**

* The deadline for grading is Monday, October 28th at 12:00pm (noon).
* 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 (Dan Holtby: djholtby@uwaterloo.ca) so that all ISA’s and TAs can hear about clarifications or changes to the marking scheme.
* 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 04 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 style portion of the code
* Unless otherwise specified, the marking scheme does not apply to bonus questions

|  |
| --- |
| **Q2c: Helper Function Use**  Students should create a helper function for counting symbols in a one dimensional list of symbols.  Errors:   * A helper function for counting symbols was not defined/used |
| **Q3a: Template Function – Book**  Template functionsshould look similar to this (there are variations in parameter naming and number of periods per ellipses which can be anywhere from 3 – 6 periods):    (define (book-template book)  (... (first book) ...  (second book) ...))  Errors:   * Missing ...s (at the beginning of the function body and the end of the function body) * Template function has more than one parameter * Missing (first book) * Missing (second book) * Using cond * Using empty? * Using (rest book) * Using recursion (i.e. (book-template (rest book)))   Exceptions:   * Students may also use (first (rest book)) instead of (second book) * Students may create their own selector functions and use them instead of (first book) and (second book). For example:   (define (title book)     (first book))  (define (author book)     (second book))  (define (book-template book)  (... (title book) ...  (author book) ...)) |
| **Q3a: Template Function – (listof Book)**  Template functionsshould look similar to this (there are variations in parameter naming and number of periods per ellipses which can be anywhere from 3 – 6 periods):    (define (listof-book-template lob)  (cond [(empty? lob) ...]  [else (... (book-template (first lob)) ...  (listof-book-template (rest lob)) ...)]))  Or  (define (listof-book-template lob)  (cond [(empty? lob) ...]  [else (**...** (first (first lob)) ...  (second (first lob)) ...  (listof-book-template (rest lob)) ...)]))  Errors:   * Missing …s (at the end of the function body, after (empty? lob), and before opening bracket after the else statement) * Template function has more than one parameter * Missing cond * Missing (empty? lob) * Missing (first lob) * Missing (book-template (first lob))/ (first (first lob)) and (second (first lob)) * Missing (rest lob) * Missing (listof-book-template (rest lob)) * Using more than 2 conditions   Exceptions:   * It is acceptable for students to use (cons? lob)instead of else * Students may also use (first (rest book)) instead of (second book) |
| **Q3b: Helper Function Use**  Students should create a helper function for inserting books into a sorted list.  Errors:   * An insert function was not defined/used |
| **Q2b: 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 |
| **Q2a, Q2b, Q3a, Q3d: Contract Correctness**  Correct contracts should be similar to the ones written at the end of this document (not including format).    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 * **Students can define their own data types and use them in the contract.** If this is the case, please look for the data type somewhere in their file (probably located at the top of their file, or use ctrl+f in the browser to find it) and make sure the data definition is sufficient enough to make the contract correct. If no data definition is provided but is used in the contract, then it is counted as an error. |
| **Q2b: 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, any kind of arrow is fine i.e. =>, -->) * Lists are not in the form of (listof x), where x is a type |
| **Q2b: 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. |
| **Q2b: Names**  Constant, parameter, and helper function names should be descriptive but not too long  Errors:   * Ambiguous names   + Note that lox (for (listof x), where x is a type) is acceptable and is not ambiguous * 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 /,?) |
| **Q2b: Code Complexity**  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 string=?)   + Students may use equal? to compare two lists * Using (boolean=? true x), (boolean=? false x),   (boolean=? x true), or (boolean=? x false)   * Code in helper/main function that overly complicate solution (Grader’s judgement) |
| **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 |

**Contracts:**

As a reminder, **students can define their own data types and use them in their contract**. If this is the case, **please look for the data definition somewhere in their file** (probably located at the top of their file, or use ctrl+f in the browser to find it) and make sure the data definition is sufficient enough to make the contract correct. If no data definition is found for the unidentified type, but is used in the contract, then it is counted as an error.

Q2a:

;; my-list-ref:(listof Num) Nat -> (anyof Num false)

Or

;; my-list-ref:(listof Num) Nat -> (anyof Num Bool)

Q2b:

;; zip: (listof Num) (listof Str) -> (listof (list Num Str))

;; requires: both lists have the same length

Or

;; zip: (listof Num) (listof Str) -> (listof (listof (anyof Num Str)))

;; requires: both lists have the same length

Or

;; zip: (listof Num) (listof Str) -> AL

;; requires: both lists have the same length

Note:

* It is acceptable if students require that the (listof Num) contains no duplicates for zip
* If students produce AL (for Association List) instead of (listof (list Num Str)), then no data definition is required for AL since it is defined in the course notes.

Q3a:

;; book-template: Book -> Any

Note:

* If students write (list Str Str) instead of Book, leave a comment but do not deduct marks (there will be an annotation for this under Contract correctness) since Book is preferred in the contract.

Q3d:

;; book-by-author?: AuthorIndex Str Str -> Bool

Or

;; book-by-author?: (listof (cons Str (listof Str))) Str Str -> Bool

;; requires: the first Str in each inner list is unique

Note:

* If students write the second contract, leave a comment but do not deduct marks (there will be an annotation for this under Contract correctness) since AuthorIndex is preferred in the contract.