**Assignment 04 – Marking Scheme**

**General Marking Notes**

* The deadline for grading is Wednesday, October 16th 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 (Charles Clarke: charles.clarke@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.
* 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 03 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

|  |
| --- |
| **Q1: Helper Function Use**  Students should create a helper function that deals with a (listof Char) otherwise they will have string->list, list->string everywhere in their code.  Error:   * Helper function for (listof Char) not defined/used   Exceptions:   * Students might solve this question using substring instead of recursing on a (listof Char). If this is the case, give them full marks in this section as they do not need a helper function if they use substring. |
| **Q3: Constant Use**  Students should create constants for each Canadian coin value to increase readability (names of constants do not have to match the ones below).  Constants that should be defined:   * (define nickel 5) * (define dime 10) * (define quarter 25) * (define loonie 100) * (define toonie 200) |
| **Q4b: Helper Function Use**  Students should call the function prime? that was created in Q4a to simplify the solution.  Error:   * prime? was not used |
| **Q4c: Helper Function Use**  Students should call the function prime? that was created in Q4a to simplify the solution.  Error:   * prime? was not used |
| **Q3: 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 similar to the ones 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). **This includes helper functions** (only deduct one rubric level for all contracts missing in **helper functions**) * 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 |
| **Q3: 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 |
| **Q3: 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. |
| **Q3: 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)   * **Defining constants for symbols** (i.e (define toonie ‘toonie)) * Code in helper/main function that overly complicate solution (Grader’s judgement) |
| **Q3: 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 /,?) |
| **Q3: 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   + **Long examples are fine**. Examples with cons tends to exceed the 80 character limit and there’s really no way to avoid it. * 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**

Q1:

;; count-vowels: Str -> Nat

Q2:

;; remove-duplicates: (listof Num) -> (listof Num)

Q3a:

;; count-change: (listof Sym)-> Nat

Q3b:

;; make-change: Nat -> (listof Sym)

Q4a:

;; prime?: Nat -> Bool

Q4b:

;; next-prime: Nat -> Nat

Q4c:

;; prime-range: Nat Nat -> (listof Nat)