This is a comprehensive marking scheme template for assignments, which includes all the general marking notes, general notes for rubrics, criteria, major errors, and exceptions for each criterion used in all the assignments in Fall 2020. It will help you understand what a marking scheme should look like, what criteria could be included and how to select appropriate questions for each criterion that is going to be checked. Notes for ISAs in blue boxes are for you to read to learn about the criterion.

**Assignment XX – Marking Scheme**

**General Marking Notes**

* The deadline for grading is Monday, due date, and time.  
  Please complete 5 submissions by Saturday, December 12st at noon so ISAs can give you feedback.
* **If you are marking more than 30 submissions, click All on MarkUs to show all the submissions you have.**
* Do not leave submissions blank. All submissions should have annotations. If a student receives full marks, leave an annotation telling them that they did a good job.
* **If you remove marks for any reason, state the reason with an annotation and highlight the most appropriate spot.**
* Reply to the AXX Official Marking Thread email if you have any questions.
* **Do not modify preset annotations**; this will change them in every occurrence for all markers.\*
* If you notice a common mistake that is not listed under the annotations, please create one under an appropriate category.
* **Do not duplicate any preset annotations**
* Remember to set each assignment’s marking status to complete when you are finished.
* Please start marking early! The earlier you start the easier it is for us to help you.
* 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 and the amount of time you spent marking.**

|  |
| --- |
| **Notes for ISAs:**  The deadline for marking should always appear the top of the marking notes.  \*All the annotations used in one submission can be viewed under “Annotations” tab. Do not edit any preset annotation there. (i.e do not change any annotations provided by ISAs that you found under “Annotations” >> ”Current Annotations”). If it is a one-time annotation, feel free to edit it if it is necessary. |

**General Notes on the Marking Rubric:**

* Unless otherwise noted, deduct one rubric level for each error.
* **Do not deduct more than one level for the same error in multiple places.**
* **If the student did not submit anything for the questions we are checking, deduct all rubric levels. \*\***
* If you notice any potential error or controversial style not covered in the rubric, consult an ISA before deducting marks.
* If a student does not submit anything for the questions we are checking (zombies), deduct all rubric levels.
* Local helper functions need to have purposes and contracts for this assignment. For example, (local [(define triple (lambda (x) (\* 3 x)))] …) needs to be documented.

|  |
| --- |
| **Notes for ISAs:**  \*\*If the student did not submit all the questions, it depends on what criterion is marking for. For example, if student submitted only 4/7 functions and no design recipe provided for the missing questions, the 3 missing contracts will be counted as 3 incorrect contracts. For other criteria, marks whatever is submitted |

|  |  |
| --- | --- |
| **QX (question-name) - Constant Use:**  Students should define the following constant value(s):  It counts as one error per bullet point below. If they are missing one or more from each category is one error per category:  (Provide required constants here.)  e.g. sinking-rate (0.05) or remaining-rate (0.95)  **Note**:   * You might think there are more values that require constants, and you might be right. These are the minimum we require from them. They can have more constants than this. * If students are not using the constants above but they have **no magic numbers** in their code, then do not deduct marks. Consult an ISA if you are ever unsure. * This section only checks the use of constants, **not how well they are named.**It is acceptable, but not required, for students to define a constant with a value of 100 for percentage conversions.  |  | | --- | | **Notes for ISAs:**  Constant use is usually checked in the first few assignments where several constants are used. Checking constant use or not does not depend on how many constants should be defined in the assignments. Sometimes there is only one constant used in the entire assignment, but it has a clear purpose, and it is something that could probably be changed. This criterion is also necessary in that case. If there are a dozen of constants, classify them into different categories according to their purposes. If constants are divided into categories, missing one or more constants from each category counts as one error. Otherwise, deduct based on the total number of missing constants. It is suggested to provide markers with required constant values beside names/purposes if the names/purpose are not intuitive enough. | |
| **Purpose Statements:**  **QX (question-name): Purpose Statements Format (Main and Helpers):**  A purpose statement should contain references to each parameter used in the function by name. References to constants are acceptable but not required. All helper functions (global or local) require purpose statements.  **Major errors:**   * Not referencing all parameters in the purpose description * Function headers at the beginning of the purpose (i.e. (fn-name param1 param 2 …)) does not exist or does not match the actual function header   **QX (question-name): Purpose Statements Correctness (Main Functions):**  A purpose statement should be clear and concise. **Select the appropriate rubric level based on how many purpose statements are correct.** Any of the following errors make a purpose statement incorrect:  **Major Errors:**   * Missing purpose (or missing function block) * Ambiguous purpose, not clearly explaining what the function does. * The purpose describes how the function works instead of what it does * The purpose only refers to another helper function (i.e. only saying this is a wrapper function of another helper function).   **Notes:**   * If purpose is correct but it is unnecessarily long and convoluted, then do NOT deduct marks, but leave a comment   **QX (question-name): Purpose Statements Correctness (Helper Functions):**  (Global and local) Helper functions should include purpose statements. **Only check for the existence of purpose statement for helper functions.** If the purpose statement for a helper function is missing, deduct all rubric levels.   |  | | --- | | **Notes for ISAs:**  statement format, correctness for main functions and helper functions belong to the overall criteria: purpose statements. It is suggested to separate these three parts in marking schemes for better clarity. “Purpose statements” is one of the criteria which are often carried forwards throughout the term. Format related issues for both main and helpers are deducted by type that is how many types of errors there are in one submission. Multiple occurrences of the same type of error count as one. By the contrast, correctness for main functions is deducted by the number of incorrect purpose statements. Since students are free to define their own helpers in the most of situations, the existence of purpose statements is the only measure of correctness for helper functions. In Fall 2020, purpose statements correctness for helpers was not an independent criterion in all the marking schemes. It was argued that students could lose all the marks for purpose statements correctness if they defined many helpers and failed to provide purpose statements for them but did a perfect job for main functions. Though students who use more helpers tend to lose more marks anyway, they will receive marks for correct answers to the best extent if the correctness for main and helper functions is marked separately. Another change worth considering is providing markers with example purposes for main functions. As local is introduced, you probably want to make it clear in the marking schemes and marking meetings that both global and local helpers will be checked for purpose statement correctness. If helper functions are exempt from including purpose statements in specific assignment, clearly notify markers as well. | |
| **QX (question-name): Examples:**  All (global) functions should have **at least 1** illustrative example to test the basic functionality of the code.   |  | | --- | | **Notes for ISAs:**  In Fall 2020, the marking schemes on this criterion were relatively lenient. Even though it was required by the style guide that “students must include each base case and at least one recursive case for recursive data”, it was not reflected in any of the marking schemes because of instructors’ opinion. The style guide is possible to change regarding examples. Primarily refer to the style guide and check with instructors if they want to strictly enforce the requirements or not. This is also one of the criteria that is carried forward throughout the term. | |
| **QX (question-name): Names:**  Constant, parameter, and helper function names should be descriptive but not too long. If students have not done the questions to be marked (that is, not name to check), **deduct all rubric levels**.  **Major errors:**   * Ambiguous names, names that are not self-explanatory * Using the value in the constant name, inflexible names (e.g. 1-num) * Violating the style guide requirement,.8 which is that names that are proper nouns like Newton should always be capitalized. Otherwise, use the Racket convention of lowercase letters and hyphens (i.e. top-tax-bracket).   **Exceptions:**   * Having names that are too long. Don’t deduct marks here but leave a comment. * Having filler words like, “the”, “that”, etc. Leave a comment if they do. * If the student uses L for a list, that is okay because we did use it in the lecture slides (although it violates the style guide) \*  |  | | --- | | **Notes for ISAs:**  Names are more frequently checked in the first few assignments while specific main functions and parameters names are usually given in later assignments. If there is very limited number of names can be checked or most of the names do not have actual meanings (e.g. lon v.s. vector), this criterion could be removed from the marking scheme. If students are doing well in naming, you could also omit this criterion in coming marking schemes.  \*theoretically and ideally, the names used in course materials should correspond with the style guide. It happened in Fall 2020 that a bad name (L for list) was used in lecture slides. Thus, it was acceptable for assignments which came after the time when the slides with the name were uploaded. | |
| **QX (question-name): Helper functions**  Students should define helper functions to avoid repeating code and enhance readability. Below are some examples of helper functions, BUT they can have different ones. Do not deduct as long as no code repetition.  (Provide examples of helper functions here.)  (e.g.  Q1a: A helper which counts the total number of students  Q1b: A helper which sums up the total grade of all the students)  **Notes:**   * If a student defines unnecessary helper functions that only performs simple calculations (e.g. add 2 numbers together), leave a note but do not deduct marks. * Students may (and will) define more helpers than listed above. Do not deduct any marks if they have more helpers. * Please consult the appropriate marking channel if you are unsure whether to deduct marks.  |  | | --- | | **Notes for ISAs:**  In general, it is hard to give specific names to suggested helper functions since students could name them very differently. The suggestion is to provide purposes of helper functions for markers. As the assignments become more complicated, the implementation of the same question is likely to vary among students. There will be no common helper functions which can be suggested. If that is the case, skip this criterion and only check code complexity if necessary. | |
| **QX (question-name): Code complexity:**   Incorrect code should still be marked for Code Complexity.  **Major errors:**   * Including a cond in the answer part of an else as below:   (cond  [(some-conditional) …]  [**else (cond**  […])])   * Using eq? or equal? instead of more specific predicate for comparison (such as = or symbol=?) \* * Using (boolean=? true x), (boolean=? false x),(false? x) * Other unnecessary code that adds code complexity (for example, if there are repeated code that can be simplified by re-factoring or grouping cases together, deduct a rubric level and leave a comment)   **Note:**   * If students use unnecessary nested cond, leave a comment but don’t deduct marks. * [else (**some-operator** (cond ...))] is not an error. It is not the same as (else (cond ...)). * [else (**cons** ...)] is not an error. * It is ok to use [else (local […])]. * If some parameters of main functions are also used in local helper functions, it is not good style to include those parameters again in the function header of local helpers if there is not recursive call. Leave an annotation, but do not deduct marks. * If you see any errors that are not on the list above or have any doubts regarding anything above, please consult the ISAs before deducting any marks.  |  | | --- | | **Notes for ISAs:**  There are some questions from assignments later in the term could be checked for code complexity. However, checking them might remarkably increase the workload of markers. Take this factor into consideration when determining to include this criterion or not.  \*If eq? or equal? is the most applicable predicate (e.g. compare two lists), it is not an error to use it in that case. | |
| **QX (question-name): Contract Correctness (Main Functions):**  Correct contracts should be exactly as 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.  **Major error:**   * Missing contract (or missing function block) * Incorrect type or incorrect number of types listed (other than the exceptions below) * Missing requirements when necessary   **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 true, but unnecessary requirements, **leave a comment,** but do not deduct marks. * Requirements can vary as long as they depict the right message (ne-listof X) is the same as (listof X) with the requirement that the list is non-empty.   **Note:**   * Students can use self-defined data types in the contracts. If they do so, check the data definitions to see if they incorporate the requirement; if they do so, they can use these types in the contract and leave out the requirement section.  |  | | --- | | **Notes for ISAs:**  This is one of the criteria carried forwards throughout the term. Contracts are important in hand-marking, especially the correctness of main functions contracts. Try to check as many questions as possible for the first few assignments. As students get more used to writing contracts, you can omit questions with very intuitive contracts and only check those with harder ones. Contract correctness for main functions is deducted by the number of incorrect contracts. In Fall 2020, sample contracts, alternatives as well as data definitions used in question to be checked at the end of marking schemes. | |
| **QX (question-name): Contract correctness (Helper functions):**  (Global and local) Helper functions should include contracts. **Only check for the existence of contracts for helper functions.** If the contract for a helper function is missing, deduct all rubric levels. |
| **QX (question-name): Contract format:**  Contracts should be formatted correctly. **If no contracts are present, deduct all rubric levels; otherwise, mark the existing contracts for all questions.**  **Major errors:**   * Missing/Incorrect function name * Surrounding function names in brackets * Including parameter names before the colon * Missing colon after function name if the function name is provided * Missing uppercase letter to begin type name (num instead of Num). Self-defined type names should also begin with capitalized letter. * Using incorrect type names (Number instead of Num) * Missing ->   **Exception:**   * If they have => instead of ->, leave a comment but do not deduct marks. |
| **QX (question-name): Whitespace/Layout:**  Solutions should be spaced appropriately for readability, and different function blocks should be separated.  **Major errors:**   * Missing separators between function blocks. Separators can be at least 2 but at most 5 blank lines, or a row of symbols (such as \*), or question headings. * Consistently more than 5 blank lines between function blocks. * Multiple lines that exceed 80 characters. * Design recipe not in order, or design recipe for main function interrupted by helper functions, constants or data definitions. Helper functions and constant definitions should be at the top or bottom of main function block but should never appear in the middle of it. * Awkward/incorrect indentation * Consistently missing spaces in their code: e.g. (and(or(cond[]))) * Lines that are too vertical. * Multiple occurrences of brackets in own lines.   **Exceptions:**   * If a contract is not right before the function definition, leave a comment but do not deduct marks. * If a student defines a long list for test cases and results in exceeding 80 characters in a line, leave a comment but do not deduct marks. * If students have excessive blank lines at the end of their files, leave an annotation but do not deduct marks.  |  | | --- | | **Notes for ISAs:**  Not all the major errors have to be included in every marking scheme. You can be more lenient with the line length of test cases when lists, structures and trees are introduced. | |