Assignment 2 Marking Scheme

This page describes how Assignment 2 will be graded, containing both a mark breakdown and some details we'll be looking for in each category. Read through this carefully before submitting - the last thing we want is for you to lose marks because you didn't know what we would be looking for!

Questions 1 & 2 Correctness: 20%

Your mark in this section will be determined solely on the percentage of tests passed for these two questions.

Questions 1 & 2 Design: 5%

A small mark will be allocated for ensuring that you use macros correctly in your submission for questions 1 and 2. Your solutions can be quite short; this mark is really just a sanity check to make sure you aren't using forbidden constructs like mutation in your solution. Documentation is not required, although it may be helpful to your TA for understanding the design of your solutions.

Question 3 Design: 10%

We are looking for three things in this part: A plausible syntax for users to define a constructor (many possible solutions). Correctly initializes attributes for public access and use by methods Can handle default values and auxiliary computation

Question 3 Example: 5%

Have you translated the given Python example into your new syntax correctly? Is your answer actually consistent with your macro, and does it have the same meaning as the original Python example?

Question 3 Documentation: 5%

A small mark will be allocated for English documentation describing the design of your macro. This can be in the form of inline comments or a block comment above your macro.

Questions 4 & 5 Correctness: 20%

Your mark in this section will be determined solely on the percentage of tests passed for these two questions.

Questions 4 & 5 Design: 5%

Like the Question 1 & 2 Design mark, this is simply a sanity check to make sure you are using the backtracking library properly. You may not use explicit mutation (e.g., `set!`) in these two questions; instead, you should rely on the backtracking API to handle getting "new choices" for you. Also, please note that your solution for Question 5 will be rather slow; it is, after all, nothing more than a brute force search algorithm. You may want to think about how to use permutation to make it faster, although this is not required for this assignment. Documentation is not required, although it may be helpful to your TA for understanding the design of your solutions.

Question 6 Correctness: 10%

Your mark in this section will be determined solely on the percentage of tests passed for this question.

Question 6 Design: 5%

We are looking for a correct implementation of `fold-<` which demonstrates your understanding of how the backtracking library is actually implemented (in particular, the `all` macro). You may use explicit mutation for this question. Documentation is not required, although it may be helpful to your TA for understanding the design of your solutions.

Question 7 Design: 10%

We are looking for two main things in your solution to this question:

- correct implementation of `peek` to view choice expressions, but not mutating the stack of choices
- making sure the public behaviour of `-<`, `next`, and all remains the same

Please note that you may reuse a lot of the existing code, but you are permitted (and in fact encouraged) to change how the above macros/functions are implemented to incorporate peek's behaviour. You may use mutation in your solution, although do keep in mind that peek should not consume any choices!

Question 7 Documentation: 5%

A small mark will be allocated for English documentation describing the design of `peek`, and any changes you made to the existing choice implementation. Please note that this means you shouldn't just document peek, but really all changes you make to the backtracking library's code.