Prelim 1: Part 2

Deadline: Friday, 10/14/16, 9:30 pm

This exam is to be done as individuals, not with partners nor with teams.

This part of the exam is "open book". Any code provided on the CS 3110 Fall 2016 website or CMS is permissible to use as part of your solution without citation.

We do not recommend that you reuse any code from your recitation solutions or your assignments. If you do reuse code from an assignment, you bear the responsibility for any bugs in it: it is entirely possible that you would get penalized on this exam as well as on the assignment for the same bugs. You must cite any such reused code in the header comment of your exam solution.

Other than those cases, you may not include source code from any other source as part of your exam solution. Doing so would consitute unauthorized aid on the examination and would expose you to charges of violating the Code of Academic Integrity.

Overview

There are three problems in the provided file `part2.ml`.

- To complete problem 0, fill out your name and netid in the provided comment, and read the Academic Integrity statement.
- To complete problem 1, finish the single type and three functions.
- To complete problem 2, finish the single function.

What we provide

In the release code on the course website you will find these files:

- A template file `part2.ml` and a minimal (extremely incomplete) OUnit test suite `test_part2.ml`. The provided template file does not yet pass that test suite, because the implementations are incomplete.
- Some additional scripts for compiling your code.

What to turn in

Submit files with these names on CMS:

- `part2.ml`, containing your solution code.

You will not be submitting your OUnit test suite.

Grading issues

- **Late submissions:** There is a special late policy for this exam given in Piazza post @1101. In summary, submissions are due by 9:30 pm on Friday, with approximately a 30 minute grace period. If you upload after 9:30 pm, you do so strictly at your own peril.
- **Environment, names, and types:** Your solution must compile and run under OCaml 4.03.0. You are required to adhere to the names and types of the functions provided in the template file. Your solution must pass the `make check` described below in Part 0, and your solution must pass the two provided public test cases without any changes to them. Otherwise, your solution will receive minimal credit.
- **Code style:** Refer to the CS 3110 style guide. Ugly code that is functionally correct will nonetheless be penalized. Take extra time to think and find elegant solutions.

Prohibited OCaml features

You may not use imperative data structures, including refs, arrays, mutable fields, and the `Bytes` and `Hashtbl` modules. Strings are allowed, but the deprecated (mutable) functions on them in the `String` module are not. Your solutions may not require linking any additional libraries/packages beyond OUnit. All non-imperative modules in the standard library (e.g., List, Set, Map) may be used.

Part 0: Sanity check

We supply the usual kind of `make check` and `make test` that we have shipped with previous assignments. It is now your responsibility to solve any issues with incorrect names and types, not the consultants'. You are being tested on your ability to solve these issues.

Part 1: The coding problems

There are two coding problems. Their specifications are provided in `part2.ml`.

The first problem asks you to determine whether one hand beats another using some rules that are similar to the game of Poker. Be sure to read the rules carefully, because they may differ from whatever variety of Poker you might have played in the past. Part of what is being tested here is your ability to correctly interpret a specification; the course staff won't be answering questions on Piazza about what it means.

The second problem asks you to determine what rooms are reachable from a starting room in an adventure game. Again, be sure to read the specification closely. As it says, a small part of your grade will be based on the efficiency of your solution.

Assessment

Your solution will be assessed on the following criteria:

- Correctness: Does it correctly pass the test cases developed by the course staff? Does it adhere to the specifications provided?
- Code quality: Is the code stylish and elegant? Is it readable by humans?

Your solution will not be assessed on specification comments or on your test suite, because the course staff simply won't have time to grade these things if we're going to attempt to get your exam scores back to you before the drop deadline. Please don't somehow infer that these things are unimportant! We are omitting them from grading only because of time constraints.