CS 3110 Coding Standards Rubric

	Exceeds expectations (1 point)	**Meets expectations (0 points)**	**Needs improvement (−1 point)**
Documentation	The `ocamldoc` extracted docs are a pleasure to read. Every name (module, structure, function, etc.) has well-written and well-formatted documentation. Markup and tags are used effectively. Example usages are provided. Better than the OCaml standard library docs.	All names have documentation in the extracted docs. Each function has at least a sentence describing its output in terms of its inputs. Similar to the OCaml standard library docs.	Several definitions are missing documentation in the extracted docs, or their documentation is unedifying or perfunctory to someone who is not the author of the code, or docs cannot be extracted because of coding errors. Worse than the OCaml standard library docs.
Testing	Extensive, industrial-quality test suite. Thorough edge and error case coverage. All functions exposed through interfaces tested.	Adequate test suite for a course assignment. Every required function has multiple unit tests demonstrating correct operation.	Test suite has not been expanded significantly beyond release code. Test suite will not compile (okay to compile and have individual tests fail).
Comprehensibility	Code is a work of art—well organized, easy to read, stylish, and beautiful. Complexity is minimized, not dense, with minimal nesting depth. Names are well-chosen, descriptive, consistent, and not verbose.	Code is mostly readable with at most a handful of minor issues. Style is okay. Names are generally well chosen. Most code is not dense. Each function body is visible on one screen without scrolling.	Code is frequently difficult to read. Complexity abounds. Style is poor. Names are not self-documenting, or are kludgy and verbose, or do not follow consistent conventions. Nesting depth exceeds two control structures (e.g., triply nested pattern match). More than one function overflows the screen.
Formatting	N/A	Code uses spaces, not tabs. Lines are shorter than 80 characters. Indentation is consistent and least as good as `ocp-indent`. Blank lines and spaces are used effectively.	Significant and pervasive formatting errors. Mixed spaces and tabs. Lines are long and unreadable in grader's editor. Indentation obscures structure of code.