

Programming Assignment I

Due Date: See online CISC 672 schedule

Project Goals:

The purpose of this assignment is ensure that you have a solid knowledge and understanding of the syntax and semantics of the Cool language. In addition, you will learn how to read and understand a language specification in order to be able to implement the language as specified. Lastly, you should gain some appreciation for the task of test case generation, in order to adequately test a software application, in our case, to test a compiler.

This assignment will *not* be done with a partner; you should turn in your own individual work. This is a fairly straightforward assignment and most students shouldn't find it too time-consuming; however, we are giving you about a week to work on it. Don't wait until the day before to start!

Project Summary:

Write 2 Cool programs. Cool is the language you will be implementing this semester. Cool shares many similarities with C/C++/Java but keep in mind that not all features exactly match. Our hope is that using familiar syntax will make things easier for you, but you'll need to keep on your toes where it is different as well.

Writing Cool Test Programs:

Read the Cool language manual, far enough to understand the lexemes, syntax, and semantics of the language. The Cool manual is on the course web page. Based on the specification, perform the following tasks:

- Program Number 1: Write a 100-200 line Cool program that is object-oriented, and tests a large number of the features (syntactic as well as semantic) of the Cool language. Your first program should not be a trivial program, but one that does something useful. You could write a program that plays some game, works as a calculator, or implements a data structure or common algorithm, or performs some other useful task. You could look though a C++ or Java book, choose an exercise, and implement it in Cool.

Program Number 2: Write a 100-200 line Cool program that is object-oriented, and focuses on testing a specific significant feature of Cool as thoroughly as possible. An example might be to test combinations of control flow, or different kinds of parameter passing. The test program should include simple, basic test segments as well as push the limit of what is allowed by Cool for that feature(s). This program does not need to do anything useful, but serve as a test program for a Cool compiler.

- You should compile your programs using the complete Cool compiler provided to you, and described in the Cool Manual. You should execute your translated Cool programs using the spim interpreter provided to you and also explained in the Cool manual. Be sure that your Cool programs are thoroughly tested and run to your specifications.
- Write documentation that describes how to compile and run your Cool programs easily, lists the features of Cool that are "covered" or included in each Cool program, and explains what each Cool

program is supposed to be performing. Be sure to include internal documentation in each program as you would any program that you are writing for someone else to follow the logic. Describe the intended input/output of each program. Be sure to describe any limitations of your Cool programs, in terms of kinds of inputs they can take, so the TA does not try those inputs.

- Write 2 test cases (preferably file input) for each Cool program. Each test case should be a legal input and the expected output from running your program with that input.

Project Submission: You should create a tarfile of all of your files and email them to the TA, by the deadline, unless otherwise specified by the TA. You should also schedule a demo with the TA to be done within 5 days after the due date.

Evaluation Criteria:

Your grade on this assignment will be based on:

30 pts Correctness and Design:

___You are to implement two 100-200 line Cool programs.

Program 1: 15 points

___Performs a useful task

___Well-written in terms of structure

___Creativity of design

___Good program structure

___Passes various input tests

___Uses alot of the features of Cool, especially oop.

Program 2: 15 points

___Thoroughly tests the specified feature

___Fairly significant feature to test

___ Passes various input tests

8 pts Documentation

___Internal

___External

5 pts Test cases

___Effective at exercising much of the Cool program

___Provided input with expected output

2 pts Ease of compile/run/test for the TA

___Easy for the TA to untar, compile, and test with given inputs