CSCI 4220 Project

1 Project Overview

In this course, you will be required to do a project. This will allow you to further develop your understanding of the material covered by applying it in a hands-on fashion. You should form groups consisting of up to four members.

Each of you will be responsible for understanding all aspects of the project, and I reserve the right to test you on this material in class.

The project will be broken down into 4 milestones, each with an associated set of deliverables.

1.1 Milestone 1

The first milestone concerns itself with defining the syntax of the language and can be accomplished by developing a context-free grammar for the language as well as a specification of the tokens that are used by that grammar. In the first milestone, you should also explicitly state the associativity and precedence of each operator in your grammar.

- Inputs to Milestone 1
  - this document
- Expected Outputs
  - A BNF description of your language.
  - A regular expression description of the tokens in your language.
  - A document describing the precedence and associativity of the operators in your language.

1.2 Milestone 2

In the second milestone, you will be asked to provide a formal denotational description of the semantics of your language. An organized way to do this is to first define a model of computation (an abstract computer) and to then describe in general terms what it means to "execute" each construct in your language.

- Inputs to Milestone 2
  - The outputs of Milestone 1.
- Expected Outputs
  - A denotational description of the semantics of your language written in the style presented in class.

1.3 Milestone 3

In the third milestone, you will construct a TL domain for your language. This domain will allow you to parse programs in your language, check for the presence of ambiguously defined constructs, view the parse tree corresponding to programs, and output the parse tree to a file. How this can be accomplished will be discussed in class. In this phase you are also required to (1) translate a small Java program that I give you into an equivalent program in your language, and (2) verify that this program can be correctly parsed.

- Inputs to Milestone 3
  - The outputs of milestone 1.
  - A sample program, which I will provide.
  - A basic (sample) TL domain, which I will provide.
- Expected Outputs
  - A zipped TL domain containing the files necessary to parse programs in your language. Please remove the bin directory from your TL domain.
  - The sample program translated into the syntax of your language.

1.4 Milestone 4

In the fourth milestone, you will be asked to write a denotational-semantic based interpreter for your language. This interpreter must be written in ML and integrated with the TL system.

- Inputs to Milestone 4
  - A template providing a basic interpreter architecture that has been integrated with the TL system.
  - The syntax folder developed in milestone 3.
- Expected Outputs
  - A TL domain in which the interpreter for your language is integrated.
  - A set of programs in your language demonstrating that your interpreter works. This set must contain your language's version of the `three_x_plus_1.java` program that was provide in milestone three.

2 Language Overview

The project involves the design of a small imperative programming language containing the following constructs:

- Statements
  - Declaration statements
  - Assignment statements
  - Conditional statements
    - if-then-else statements must be supported in your language

    - if-then statements must be supported in your language (be careful of the dangling else problem).
  - Pre/post increment/decrement statements (e.g., $++x;\ x--$; etc.)
  - Iterative statements
    - while-loops must be supported
    - for-loops must be supported
  - Block statements
  - Print statements – a statement that prints the value of an expression

- Expressions - your language should support the full range of expression involving the operations shown below. Note that your syntax will need to include parenthesis.
  - Types: Boolean and Integer
  - Operations:

Description	**Operator**		**Signature**
conjunction	and	:	boolean $*$ boolean $\rightarrow$ boolean
disjunction	or	:	boolean $*$ boolean $\rightarrow$ boolean
negation	not	:	boolean $\rightarrow$ boolean
unary minus	$-$	:	integer $\rightarrow$ integer
post-increment	$++$	:	integer ref $\rightarrow$ integer (has a side-effect)
pre-increment	$++$	:	integer ref $\rightarrow$ integer (has a side-effect)
post-decrement	$--$	:	integer ref $\rightarrow$ integer (has a side-effect)
pre-decrement	$--$	:	integer ref $\rightarrow$ integer (has a side-effect)
add	$+$	:	integer $*$ integer $\rightarrow$ integer
subtract	$-$	:	integer $*$ integer $\rightarrow$ integer
multiply	$*$	:	integer $*$ integer $\rightarrow$ integer
integer divide	div	:	integer $*$ integer $\rightarrow$ integer
modulus	mod	:	integer $*$ integer $\rightarrow$ integer
exponent	$\hat{}$	:	integer $*$ integer $\rightarrow$ integer
abs	$\lvert\ \rvert$	:	integer $\rightarrow$ integer
less-than	$<$	:	integer $*$ integer $\rightarrow$ boolean
greater-than	$>$	:	integer $*$ integer $\rightarrow$ boolean
equal-to	$=$	:	any $*$ any $\rightarrow$ boolean
not equal-to	$!=$	:	any $*$ any $\rightarrow$ boolean