# CS 4850 Fall 2014 Flex Assignment

Due day of the exam at 10:30 pm. Good to do before the exam as the exam will assume you understand flex and how it interfaces to other C code, difference between token and value etc.

## Objectives

* Learn to use a state machine generator.
  + Regular expressions in C code out
* Start on the semester’s project a language interpreter
* Set up way to debug what the lexical analyzer is returning to the higher level parser

## Basic Plan

Use the program Flex to generate a C file with a state machine which parser regular expressions and executes your C code for each match. “lex.yy.c”

Write a C program which is compiled along with the output C file (lex.yy.c) from Flex which tests the parser.

<http://flex.sourceforge.net/manual/>

Your test program will take the place of the file generated by Bison/Yacc which you will use later.

Your program will loop

Call yylex() which will return an integer which should be a Token constant defined by you

You will print a string version of the Token and the value lex matched

So let’s say you have defined INTEGER as 23

The flex below, will if it reads “123”

[0-9]+ return INTEGER;

return 23

With the string 123 stored in yytext

Your program will use the 23 to look up the string “INTEGER” in an array of strings

And print: Token INTEGER, Value 123

## Details

You should have a grammar that returns a different token for each of the following

Negative or positive

Float (examples -123.0 or 123.0) no +123.0

Integer (examples -123 or 123)

The binary operators each with different token

+ addition, - subtraction, \* multiplication, / division, % modulus, ^ raise to power

< less than, <=, >, >=, == equal to, != not equal to

= assignment

Unary operators

! Boolean not

Key word symbols

(, ), {, }

Skip anything else

### yytext vs yyval

Your choice which you use.

Yytext will have the string lex matched as in the example above. This makes your test program simple since you only have to look up the token’s string in an array indexed by the return value and print yytext.

If you use yyval you will need a switch on the return value of yylex() which prints the token’s string and the value from the correct field of yyval.

Bison requires you to use yyval which is a struct defined as well as the TOKEN values at the top of bison with a syntax like:

%token <floatval> FLOAT

%token <intval> INTEGER

Which generates a struct yyval{ int intval; double floatval; }

In lex your grammer would be

[0-9]+ yyval.intval = atoi(yytext); return INTEGER;

You are welcome to define yyval and use it in your flex so you do not have to change flex later when you start using Bison

## Rules

Define your token constants (an optional yyval) in a header file named

y.tab.h

which is included at the top of your program and lex.yy.c

This is the file that Bison generates if you use the –d switch.