CST 334: Operating Systems

Dr. Glenn Bruns

# A very simple syntax checker

**Purpose.** The purpose of this homework is to help you get some first experience in writing a parser, which is a program that checks whether an input string has valid syntax. We will write C code that checks if an input string can be derived from a simple BNF grammar.

**Instructions.** The idea is that I provide you with a very simple parser named parens.c. Your job is to complete a very similar parser named blist.c that's based on a slightly-different BNF grammar.

1. Copy the files parens.tar and blist.tar that can be found in mlc104 at /home/CLASSES/brunsglenn/cst334/hw/hw11 to a directory of your own.
2. Extract the files from parens.tar. The tar file contains parens.c, a Makefile, and some test scripts.
3. File parens.c checks whether an input string can be derived from this simple BNF grammar:

expr ::= a | ( expr )

Strings that can be derived from this grammar are "a", "( a )", "( ( a ) )", "( ( ( a ) ) )", etc. The string "( a" cannot be derived from the grammar.

1. Look at the Makefile and run 'make' to run the test scripts. They should all pass. You can also test parens manually with a bash command like this:

echo '( ( a ) )' | ./parens

In this case, the output will be 'yes' because '( ( a ) )' can be derived from the grammar.

1. Spend some time reading and understanding parens.c. Do not rush this step. The key functions in parens.c are lexan(), that splits the input into pieces called "tokens", and expr(), that checks to see if the input can be derived from the BNF grammar. As an example to understand lexan(), if the input is "( a )", then the tokens are "(", "a", and ")".
2. Extract the files from blist.tar, and look at blist.c. At the top of the file you can see the grammar it uses:

expr ::= a | b "," expr

Strings that can be derived from this grammar include "a", "b , a", "b , b , a", etc.

1. Edit blist.c to replace the comment // YOUR CODE HERE with code so that blist.c correctly checks whether its input is a string that can be derived from the grammar. Do not modify the code except to replace this comment.
2. blist.tar also contains a Makefile and test scripts you can use for testing.

Submitting. Submit your edited blist.c on iLearn.

Grading. 6 tests will be run on your code. Some of these tests may differ from the ones provided in blist.tar. 10 points for each test that passes.