ANKARA UNIVERSITY
Computer Engineering
COM241 Programming Languages
Lab. Assignment 1-2-3

About this assignment

This document provides you with the details of your fourth lab assignment for the COM240 Programming Languages course. This assignment is double weighted (it is worth two times the worth of previous lab assignments). This is an individual assignment. However, you are encouraged to discuss the technical approaches with other students.

Description of this project

The task is to obtain a lexical analyzer and parser for a language called small-C, which is a simplified subset of C. You will be using *flex* (a lexical analyzer generator) and *bison* (a parser generator upwards compatible with *yacc*) for generating the compiler. In the end, you should acquire a parser which can check the syntax of a source program (whether it is wellformed
or not) and (in either case) terminate.
Note that some legal C programs are not legal programs for small-C.

The languages involved

The syntax for small-C is given at the end of this document. What you mainly have to do is to translate the given syntax diagrams to flex and bison input. Be careful not to change the language! You must also handle C comments.
Small-C only has two types of variables: integer and character. A *string* is a (possibly empty) sequence of characters (excluding newline) between double quotes, e.g. "small.h". A *character* is a C character of the form 'c' where c is a single keyboard stroke, or \n, \t, \\, \' or EOF (end-of-file). You don't have to handle all possible C character definitions, for example '\045'. A *number* is a sequence of digits. A number cannot be empty. An *ident* (short for identifier) is a non-empty sequence of characters [a-zA-Z0-9_] beginning with an alphabetic character. A variable must be declared (exactly once) before used. All variables get the initial value 0 or '\000'. Binary subtraction, addition, multiplication, division and modulus operations are all left-associative. So 5 – 3 – 2 evaluates to 0, not 4. Unlike the C you know, they can only be applied to integers, not characters. Conditions in *while* and *if* statements are interpreted as follows: 0 stands for false and any other value stands for true. An else binds to the closest previous unbound then. The #include lines should be ignored by your compiler, they are included so that the usual C compiler will also work on the input. The `read` and `readc` procedures read an integer and character respectively, while `output` and `outputc`
output an integer and character respectively. You must handle non-recursive procedures and functions. You should check that each variable or function is defined before it is called.

Infrastructure

- flex http://flex.sourceforge.net/
- bison http://www.gnu.org/software/bison/
- yacc http://dinosaur.compilertools.net/yacc/
- Wikipedia Entry for Small-C http://en.wikipedia.org/wiki/Small-C

Assignment

The assigned work is to develop a front end part of a compiler for the small-C language. You will develop a lexer (lexical analyser) and parser for the small-C language based on the flex A specification of the small-C language can be found at the end of this document. In a second step, you will develop a parser for the same language using bison. At the end, the program

you have should be able to process a given file and determine whether the contents of the file form a valid small-C program or not.
Bonus: You will get extra points if your program can provide the user with meaningful error messages before terminating.
Hints:
• You can find many flex and bison examples online. You should spend considerable amount of time to go through such examples.
• Write a couple of small Small-C test programs to test your program. Include correct and incorrect Small-C programs and check whether they are correctly accepted or rejected.

Deadlines and submission

The deadline for completion of the assignment is

Due Data: Tuesday, November 22th, 2016
You will demonstrate your program to the course assistant and answer some questions.

Appendix