ECE/CPSC 3520
Fall 2015
Software Design Exercise #1

Blackboard submission only
Assigned 9/4/2015
Due 9/25/2015 10:00PM

Contents

1 Preface

The overall objectives of SDE1 are:

1. To gain experience in designing, implementing and testing scanners and parsers.

2. To introduce flex and bison as general purpose scanner/parser development tools.

3. To use flex and bison to build a grammatical recognizer to distinguish between 5 languages (grammars), or 6 cases if we include 'none of the above'. These grammars will require you to consider and implement contextual constraints.

   Alternately, we are designing, implementing and testing a syntactic recognizer, implemented as a scanner/parser.

4. To give you experience in grammatical inference, i.e., given a specified $L(G_i)$, you will design G_i and then implement a scanner/parser for it using flex and bison, respectively. Grammatical ambiguity is not desired.

All work is to be done on a linux platform. Your grade is based upon a correctly working implementation. Significant in-class discussion will accompany this SDE. **Get started now.**[1]

This is an individual (not group) effort.

2 Resources

As discussed in class, it would be foolish to attempt this SDE without carefully exploring:

1. The text, especially the many examples in Chapter 6;

2. The `flex` manual (`http://flex.sourceforge.net/manual/`); and

3. The `bison` manual (`http://www.gnu.org/software/bison/manual/`).

[1]At least install flex and bison.

3 The 5 Languages (6 Cases)

First, consider the alphabet:

$$V = \{a, b, c, d, e\}$$

Now the language/grammar cases:
Case 1:

$$L(G_1) = \{a^i b^j c^k d^l\} \text{ with } i, j, k, l > 0 \text{ AND } ((i \neq j) \text{ AND } (j \neq l))$$

Case 2:

$$L(G_2) = \{a^n b^n c^n\} \text{ with } n > 0$$

Case 3:

$$L(G_3) = \{a^n b^n c^n d^n\} \text{ with } n > 0$$

Case4:

$$L(G_4) = \{a^n b^k c^n d^k\} \text{ with } n, k > 0 \text{ AND } n \neq k$$

Case 5:

$$L(G_5) = \{c^n d^{n+1} e^{n+2}\} \text{ with } n > 0$$

Case6:
'None of the Above'. The string is not in any of the languages described above. See the examples.

4 Using `flex` and `bison` to Determine The Language/Grammar (or Class) of an Input String

Here you will treat scanning and parsing of an input file as if it were 'source code' constrained by one (or none) of the grammars whose corresponding languages are shown in Section 3. I'll discuss syntactic correctness in class and show examples of correct and incorrect files.

This SDE MUST be done using both `flex` and `bison` (and c).

5 Naming and Building

The evaluation of your work will be partially automated. Therefore, you must adhere to a number of specifications. The `flex` input file is to be named `whichclass.in`; the bison file is to be named `whichclass.y`; the `c` file containing `main` is to be named `whichclass.c`; and the linux executable is to be named `whichclass`. We will use `flex` and `bison` to build your executable with the following script (`buildit`):

```
#! /bin/bash
## to build scanner/parser */
bison -d $1.y
flex $1.in
gcc $1.c -lfl -Wall -o $1
```

For reference, here is the building of my solution:

```
$./buildit whichclass
$
```

Anybody see any errors or warnings?

6 Sample Use

Here are a few cases. I'll discuss these in class. Your effort should follow the output format shown after '`grammar recognizer result:`'.

```
NOW TESTING PARSER GRAMMAR RECOGNIZER

****************************************************
test file name: aaabbbbcccdddd
test file contents:
aaabbbbcccdddd
grammar recognizer result:

>>>> Grammatical Recognizer (SDE1) <<<<<<
Grammar G4 Recognized
****************************************************


****************************************************
test file name: aaabbbcccddd
test file contents:
aaabbbcccddd
grammar recognizer result:

>>>> Grammatical Recognizer (SDE1) <<<<<<
 Grammar G3 Recognized
*****************************************************


*****************************************************
test file name: aaabbc
test file contents:
aaabbc
grammar recognizer result:

>>>> Grammatical Recognizer (SDE1) <<<<<<

'Sorry,input string not in any of the recognized grammars (None of the Above)'

*****************************************************


*****************************************************
test file name: aaabbcd
test file contents:
aaabbcd
grammar recognizer result:

>>>> Grammatical Recognizer (SDE1) <<<<<<
Grammar G1 Recognized
*****************************************************


*****************************************************
test file name: aabbcc
test file contents:
aabbcc
grammar recognizer result:

>>>> Grammatical Recognizer (SDE1) <<<<<<
Grammar G2 Recognized
*****************************************************


*****************************************************
```

```
test file name: cccddddeeeee
test file contents:
cccddddeeeee
grammar recognizer result:

>>>> Grammatical Recognizer (SDE1) <<<<<<
Grammar G5 Recognized
*****************************************************


*****************************************************
test file name: ccddee
test file contents:
ccddee
grammar recognizer result:

>>>> Grammatical Recognizer (SDE1) <<<<<<

'Sorry,input string not in any of the recognized grammars (None of the Above)'

*****************************************************


*****************************************************
test file name: cddeee
test file contents:
cddeee
grammar recognizer result:

>>>> Grammatical Recognizer (SDE1) <<<<<<
Grammar G5 Recognized
*****************************************************


*****************************************************
test file name: cddfeee
test file contents:
cddfeee
grammar recognizer result:

>>>> Grammatical Recognizer (SDE1) <<<<<<

'Sorry,input string not in any of the recognized grammars (None of the Above)'

*****************************************************
```

```
*****************************************************
test file name: ffddee
test file contents:
ffddee
grammar recognizer result:

>>>> Grammatical Recognizer (SDE1) <<<<<<

'Sorry,input string not in any of the recognized grammars (None of the Above)'

*****************************************************
```

7 Notes

- Building of the scanner parser should not generate any shift/reduce conflicts and/or useless rule or nonterminal warnings.
- The parser should never report that a file is syntactically correct in addition to reporting an error.
- If we can't build your scanner/parser with the `buildit` script, we can't grade it.

8 Format of the Electronic Submission

The final **zipped** archive is to be named **`<yourname>`-sde1.zip**, where `<yourname>` is your (CU) assigned user name. You will upload this to the Blackboard assignment prior to the deadline.

The use of `gcc`, `flex` and `bison` should not generate any errors or warnings. The grader will attempt to build and then test your executable. The grade is based upon a correctly working solution.

The minimal contents of your archive are:

1. A `readme.txt` file listing the contents of the archive and a brief description of each file. Include 'the pledge' here. Here's the pledge:

Pledge:
On my honor I have neither given nor received aid on this exam.

This means, among other things, that the code you submit is **your** code, *and it was developed independently.* Thus, you should not be discussing your approach or solution with others in class or anywhere else.

2. The `flex`, `bison` and `c` sources for your implementation named `whichclass.*`, where * is `in`, `y` and `c`, respectively. **Do not provide an error handling routine – we will use the `yyerror.c` file shown in class and in the book, page 180, Figure 6.11 to build the executable.**

3. A log of 10 sample uses of the resulting executable (both success and failure) for different classes. Name it `whichclass.log`.