CS 335 Semester 2019–2020-II: Assignment 2

29th January 2020

Due Your assignment is due by Feb 12 2020 11:59 PM IST.

General Policies

- You should do this assignment ALONE.
- Do not plagiarize or turn in solutions from other sources. You will be PENALIZED if caught.
- We MAY check your submission(s) with plagiarism checkers.

Submission

- Submission will be through Canvas.
- Create a zip file named "`cs335_<roll>.zip`". The zipped file should contain a folder `assign2` with the following files:
  - Submit a `PDF` file with name "<roll-no>.pdf".
  - Implementation files in your chosen implementation language. Include compilation and execution instructions in the `PDF` file. Also, describe any tools that you used.
  - You SHOULD USE LaTeX typesetting system for generating the `PDF` file.
- Submitting your assignments late will mean losing points automatically. You will lose 20% for each day that you miss, for up to two days.

Evaluation

- Please write your code such that the EXACT output format is respected (if any).
- We will evaluate your implementations on a Unix-like system, for example, a recent Debian-based distribution.
- We will evaluate the implementations with our OWN inputs and test cases, so remember to test thoroughly.

Problem 1 **[50 points]**

For the following grammar, design a predictive parser and show the predictive parsing table. Remove left-recursion (if any), left-factor your grammar.

$$S \to (L) \mid a$$
$$L \to L, S \mid LS$$

Problem 2 **[50 points]**

Show that the following grammar is LALR(1) but not SLR(1).

$$S \to Lp \mid qLr \mid sr \mid qsp$$
$$L \to s$$

Problem 3 **[50 points]**

Construct an SLR parsing table for the following grammar.

$$R \to R \mid R$$
$$R \to RR$$
$$R \to R^*$$
$$R \to (R)$$
$$R \to a \mid b$$

Resolve the parsing action conflicts in such a way that regular expressions will be parsed normally. Include your disambiguation rules in the `PDF` file.

Problem 4 **[50 points]**

A dissertation consists of a title and one or more chapters. Each chapter consists of zero or more sections. A section consists of one or more paragraphs. Each paragraph can consist of one or more sentences. Sentences can be of three types: declarative, exclamatory, and interrogative. Declarative, exclamatory, and an interrogative sentence ends with '.', '!', and '?' respectively.

- You can assume that "Title", "Chapter", and "Section" are keywords, and will not appear in the text body.
- A paragraph may start immediately after a Chapter or Section. Two paragraphs are separated by at least one blank line.

Punctuation marks are period, comma, exclamation, semicolon, and question mark. Sentence separators are only period, exclamation mark, and question mark. Whitespace, comma, and semicolon are valid separators between any two words within a sentence.

You can assume the words will be well-formed from the English alphabet.

The text may include decimal numbers. Note that integers and floating point numbers (only using decimal point, no scientific notation) may appear in a sentence in a paragraph.

We have provided a sample dissertation to elaborate on the specifications. Write a parser for such a semi-structured dissertation. Compute the following statistics.

1. Print the title of the dissertation.

2. Print the total number of chapters, sections, paragraphs, sentences, and total number of words across all paragraphs. That is, ignore the title, chapter, and section lines while counting words. Also, ignore digits and numbers.

3. Print the number of declarative, exclamatory, and interrogative sentences.

4. Pretty print a Table of Contents (ToC) for the dissertation. You should limit till Sections in the hierarchy.

Print in the following format:

```
Title: An Elementary Introduction  to Compiler Design
Number of Chapters: 7
Number of Sections: 9
...
Table of Contents:
Chapter 1: Introduction
    Section 1.1: Contribution One
    Section 1.2: Contribution Two
    Section 1.3: Contribution Three
Chapter 2: Background
Chapter 3: Contribution One
    Section 3.1: Introduction
    ...
```

You are free to use any parser generator (for e.g., Yacc, Bison, or ANTLR) for your implementation. Remember to include the source files and instructions in your submission.