CS 315
Project 2

Assigned: Nov. 1, 2021
Due: Nov. 8, 2021 23:59

Parser for a Programming Language for Drones and its Lexical Analyzer

The second project builds on your language design of the first project. This project involves building a parser for your language using the `yacc` tool. Please refer to the description of Project 1 for the requirements for your programming language design. There are some minor changes, please read carefully the instructions below.

Part A - Revised and Augmented Language Design (20 points)

The requirements for the language are the same as Project 1 except for a few minor extensions. You can use as much of your previous design work as you can. However, if you have not already done so, you should incorporate the following elements into your design for the second part of the project:

- Main program statements (beginning of the execution),
- Declarations (variables, constants etc.),
- Assignment statement,
- expressions (arithmetic, relational, boolean, their combination),
- precedence, associativity of the operators,
- Conditional (selection) statements,
- Loop Statements,
- Statements for input / output (to ask questions and read answers),
- function definition and function call statements.
- Comments.

Please note that there is no single correct answer. This is a design project. As long as your language is consistent, unambiguous and it makes sense with respect to the specifications given above, it is fine. However, it is expected to be readable, writable and reliable, as much as possible.

Part B - Implementing the Parser (60 points)

For the second project, you are required to implement a parser using the `yacc` tool. The parser reads the source code of a program, written in your programming language from an input file. If the source code represents a valid program in your programming language, the parser should print out a message indicating the acceptance of the input (e.g. "Input program is valid"). Otherwise, the parser should print out an error message indicating the line number of the source code that contains the error (e.g. "Syntax error on line **!" where ** will be the line number of the source program at which the error was detected).

You should use the lexical analyzer that was developed in the project, but you may have to modify it; for example, to count line numbers. Also, the lexical analyzer will return tokens, instead of printing messages.

VERY IMPORTANT NOTE:

- Your `yacc` and `lex` specification files must compile in the `dijkstra.cs.bilkent.edu.tr` machine; otherwise, you will receive 0 from Part B.
- You should strive to eliminate ALL conflicts and ambiguities in your language, modifying your grammar if necessary. You will need to provide unquestionably convincing arguments for any conflicts that are left in your final submission.

Part C - Example Program (10 points)

Finally, you will write two programs in your language for spraying a chemical to two different shaped fields, say one is triangular and the other is rectangular shaped. You can choose the length of each edge of the fields and encode them into your program. Your program should drive the trejectory of the drone completely covering the field. You may assume that the drove will start flying in a point inside the field.

Part D - Teamwork (10 points)

You will be working with the same group you worked for Project 1. Since this is a team project, each member is expected to put about the same amount of work into the project. However, sometimes this is not the case. The remaining 10 points of your grade will come from the peer assessment. Each member will evaluate him/herself and the other members of the team. We will use the PeAs (Peer Assessment) tool for peer evaluations and self-evaluations to assess how effectively each member contributed to the team. You will receive an email message from the PeAs system that will let you enter your evaluations. You should enter your evaluations just before submitting your project. If you did not submit your evaluations to the PeAs system, your teamwork grade will be 0. The teamwork grade of a team member will be computed by taking into account the comments written and the grades received. Keep in mind that the most important part of your evaluations is the comments section.

Logistics

There are two parts that you will hand in before the due date of the project.

1. A project report (in PDF format) including the following components:
   - Title page with your **group name and ID** as well as **names, IDs and sections** for all of the project group members.
   - The complete BNF description of your language (based on the terminal symbols returned by your lex implementation)
   - General description of the structure of your language and those nonterminals that you think are important. Try to make the life of the grading assistant as easy as possible by making sure that somebody reading your report can understand and parse through a program written in your language. Make sure to note all rules adopted by your language (i.e. precedence rules and other ways in which ambiguities were resolved).
   - Description of how each nontrivial token is used in your grammar.
   - A thorough explanation of every conflict left unresolved in your final submission. Ideally, you should strive to eliminate all conflicts with no warnings or conflict errors given by yacc on your specification file.
2. Your lex and yacc description files, together with the example programs described above, written in your language. Specifically, do the following:
   - Create a folder named **CS315f21_teamXX** where XX will be your group number.
   - Copy the following files into this directory:
     - The project report (in PDF format).
     - **CS315f21_teamXX.lex** : Your lex specification file.
     - **CS315f21_teamXX.yacc** : Your yacc specification file.
     - **CS315f21_teamXX.test** : Your example program.
     - a **Makefile** that produces your complete parser with an executable called **parser** in the `dijkstra.cs.bilkent.edu.tr` machine. Check that when you type **make** in the same directory the desired executable is generated.
     - Before you proceed with the next step, you should delete all other files in this directory using the Unix **rm** command. BE CAREFUL, do not remove your lex and yacc files. MAKE FREQUENT BACKUPS.
   - Compress this folder into a single file using tools such as zip or rar.

Submission

Please upload the zip (or rar) file you created to Moodle (CS 315 (All Sections) Programming Languages) before the due date. *Late submissions will be accepted, with 20 points (out of 100) deduction for each extra day..*

If your submission does not adhere to the above guidelines, points will be deducted.
Make sure you have correct file naming.
Your parser must compile and run on `dijkstra.cs.bilkent.edu.tr`. The evaluation of your parser will be done only on this machine.
