CS 315
Project 2

Assigned: March 1, 2024
Due: March 10, 2024 23:59

Parser for a Programming Language for IoT Devices

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:

- Specification of the beginning of the execution,
- Declarations (variables, constants etc.),
- Assignment statement,
- Expressions (involving arithmetic operations, relations, boolean operations, their combination),
- Precedence, associativity of the operators,
- Conditional statements,
- Loop Statements,
- function definition and function call statements,
- Mechanism for a connection to a given URL,
- 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 (50 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 on 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 (25 points)

Finally, you will test your parser on the programs that you submitted in the first part of the project. Also show that your parser finds syntax errors by introducing small errors in these programs.

VERY IMPORTANT NOTE: If you do not submit a test program, we will have no way of evaluating your parser, hence you will receive a 0 from Part C!

Part D - Teamwork (5 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 5 points of your grade will be determined from your contributions to the project. Each member will evaluate him/herself and the other members of the team. Please fill in the form available here and email it to `guvenir@cs.bilkent.edu.tr`, on the same day of submission. Rename the files as `CS315_S24_Team_`*`X`*`-ID_`*`Y`*, where *`X`* is your team number and *`Y`* is your ID number. If you do not submit this form, your teamwork grade will be 0. The teamwork grade of each student will be computed separately, by taking into account the comments written by the student and other team members. The numerical values in columns D-H of the form will be used

in statistical analyses. Needless to say, do NOT collaborate in preparing and submitting your peer assessments.

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 you 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 **`CS315_S24_Team_X`**, where *`X`* will be your team number.
   - Copy the following files into this directory:
     - The project report (in PDF format).
     - **`CS315_S24_TeamX.lex`** : Your lex specification file.
     - **`CS315_S24_TeamX.yacc`** : Your yacc specification file.
     - **`CS315_S24_TeamX.txt`** : Your example program, in text format.
     - 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.
