CENG 442 Project (Part 2)

Assigned: April 3, 2015
Due: May 3, 2015

Implementing a Parser for a Robot Programming Language

The second project for this year builds on your language design from the first project and involves building a parser for it using the yacc tool. Please refer to the description of Project Part 1 for the requirements for your programming language design.

Important deadlines

May 3, 2015 23:59pm: Project due.

Part A: Revised and Augmented Language Design

The requirements for the language to be recognized by your parser are the same as Project Part 1. You can use as much of your previous design work as you can. Even though you should not make substantially major changes to your language definitions, you can make minor adjustments to increase the implementability of the associated parser.

Note, once again, that the terminal symbols in your grammar should be the tokens defined by your lexical analyzer.

Part B: Implementing a parser

For the second part of the project, you are required to implement a parser using the yacc tool. The parser reads the source code of a program written in your robot programming language from the input stream. If the source code represents a valid program in your language, the parser returns a message indicating the acceptance of the input. Otherwise, the parser should return an error message indicating the line number of source code that contains the error.

You should use the lexical analyzer that was developed in the first part of the project, but you may have to modify it (for example, to count line numbers).

IMPORTANT NOTES:

- You should eliminate as many of the conflicts as possible in your yacc implementation. If your yacc source generates any conflicts, you should explain how the default choice adopted by yacc to resolve the conflict is appropriate for your language.
- As in the first project, you are not required to write an interpreter or compiler for your language. Any test or example programs you will submit do not need to be executed by anything. We will manually go through your language description and example program to check for their correctness. All your parser needs to do is to detect whether its input is a valid program in your language or not based on your grammar definition.

Part C: Example Program

Also, in order to help us evaluate your parser, please revise and submit your test program from Project Part 1 (submission instructions below). Make sure the example program is correctly recognized by your parser.

Logistics

- You will be working with the same groups you formed for Project Part 1.
- There are two parts to what you will need to hand in at the end of the project's allowed time.
    1. A project report including the following components:
        - Name, ID and section for all of the project group members.
        - The name of your language.
        - The complete (revised) BNF description of your language.
        - Brief explanation of only the new features and differences from your Project 1 language design.
        - Any yacc implementation details that you think are important. Keep it brief and to the point, only include things that are not obvious from the BNF description, including any conflicts generated by your yacc source and how yacc's default choice is appropriate for proper semantics of your language.

    2. Your lex and yacc description files, together with the example program described above, written in your language.

Here's how you will hand in your project:

Please email your project report and program files to me (melih@cankaya.edu.tr) before May 3, 2015 23:59pm. PDF format is preferred for the project reports. Late submissions will not be accepted.