Project: Compiler and Virtual Machine for a Programming Language

Overview

Everyone has their favorite programming language and their little gripes about the language. This project is your chance to try and do better! You are to design, implement, and demonstrate your own language. You are to implement a lexical analyzer, parser, and runtime environment for a simple programming language. This is a team project and is to be completed in team of three or four students.

Outcomes Addressed

This assignment addresses the following course outcomes:

- Communicate, apply and evaluate tools, activities and artifacts involved in programming language design, translation and execution. Working with program compilation using tools such as Flex, Bison, ANTLR or hand-constructed lexical analyzer and parser using Definite Clause Grammars (DCG), to recognize and translate a language into an intermediate form (abstract syntax tree or byte code).
- Understanding and design of language runtime environments, especially accessing user-defined program values (primary data), managing runtime storage, and fundamental operations needed to implement a simple programming language.
- Using regular and context-free language specifiers (regular expressions and context-free grammars) and recognizers.

Language Constraints

Here is a list of constraints that your language must satisfy:

1. The language you design must have **operators** and **primitive types** for at least
    a. **Boolean** values (must support **and**, **or** and **not** operators)
    b. At least one **numeric type** (int, float, double, for example) (must support addition, subtraction, multiplication and division)
    c. It must also support **string** value assignments to variables. Operations on strings are optional.
2. The language must have a way to associate a value with an identifier, therefore you need to have an **assignment operator**.
3. The language must support following conditional constructs:
    a. A **ternary operator** (eg: ‘ **? :** ’ in Java)
    b. Traditional **if-then-else** construct.

4. The language must support the following looping structures
    a. Traditional **for** loop
    b. Traditional **while** loop
    c. '**for** i **in range(**2,5**)**' which should behave same as 'for (i=2; i<5; i++)'.
5. You must have a construct **print** for displaying the identifier values. It should output all the datatypes supported by your language including strings and booleans.
6. You must use open source and freely available tools to develop and build your project. You may use cmake, qmake, ant, make, or whatever is most appropriate given the tools you used. No proprietary tool must be required to compile and run your project.
7. Your compiler and interpreter should be invoked from the command-line with a single command line argument. For example (for a language named Enigma):
   ./obj/enigma
   Data/Factorial.enigma or
   java -cp classes ser502.EnigmaC
   Factorial.enigma if it's written in Java.

Grading Criteria

You will have a single grade on this assignment, that is based on the success of your software project, your documentation and your video demonstration. In grading the project, the following criteria will be considered:

- **Lexical analyzer**. Rules defined in the input to flex or your lexical analyzer, and consistency with the parser.
- **Parsing**. Does the grammar exist and accurately define the language? Do the grammar and lexical analyzer work together appropriately?
- **Runtime**. Do the parser and runtime environment agree on the intermediate form (abstract syntax tree or byte code). Does the runtime properly execute producing a correct result for each sample program?
- **Documentation**. Does the doc directory contain the PowerPoint presentation that describes in detail the language/grammar/compiler/intermediate form/runtime, and the snapshots of the demonstration of the language?
- **Video**: Does the YouTube video demonstrate the language right from the grammar, installation and the execution of the sample programs. Does each team member speak in the demonstration? (Around 3-5 seconds of introduction of your name along with the face in the video needs to be shown before each team member starts the discussion / demonstration. This helps us verify the identity of the speaker).
- **Quality and Collaboration**: The quality of your language, code, presentation, video, and the extent to which all team members have contributed to the final product, demo and video.

The Instructor/Graders will be implementing the project and reading the code in detail to verify if the demo shown in the video meets the actual code and implementation.

GitHub / Code repository

Your team must create a git project on GitHub (**private repository shared with the Instructor/Graders with read-only access**) for version control and collaboration on this project. GitHub Education provides free private repository. All the team members need to click on Get benefits for students to get the free private repository by using your existing GitHub account (if registered with .edu) or a new account (created with asu.edu). Every member must make at least two significant commits with the code to the project per week (in the phase three of the project). If the project log shows a lack of commits or a tendency to commit at the last minute then you will lose points (please note that committing ReadMe, comments, minor formatting changes, etc. will not be considered significant). Your GitHub repository should have the following structure:

- **src directory:** Source directory src that includes subdirectories compiler and runtime. Include the language source files in these sub-directories. If you use tools such as flex or bison, include their input files in these directories. Make sure that all the source files have comment headers indicating the authors, purpose, version and date.
- **data directory:** This directory should contain at least 5 sample source programs in your language. The sample programs should cover the constraints listed above in the instructions.
- **doc directory:** A PDF copy of your PowerPoint presentation that provides complete language description and intermediate form definition. A text file named contribution.txt which details the specific individual contribution that each member of the team has made in the project. This file should be updated regularly over the duration of the project.
- **readme.md:** Your ReadMe must include the following:
    - System on which your compiler and runtime are built (GNUstep, Linux, Unix, Windows or Mac OS)
    - Tools used
    - Directions/instructions to install your language
    - Directions/instructions to build and run your language (compiler/runtime).
    - ONE LINE of bash script that builds the compiler and ONE LINE that runs the runtime.
    - Link to the YouTube video

Milestones & Deliverables

Important: Only one submission per team

Please see below the milestones and the submission details for the project:

- **Milestone 1[Due: 4/9]**:

    - **GitHub Repository URL**: Create a GitHub **private repository**. Include your teammates as well as the Instructor/TA (ajay.bansal@asu.edu, sarthak.tiwari@asu.edu). Name your GitHub repo "SER502-Spring2020-Team<#>" (for e.g.: SER502-Spring2020-Team5). Team Numbers will be provided to you after all the students have submitted their team information on

Canvas. Your team must submit the link to your team's repository on canvas. This repository will be monitored by the Instructor/Graders to see your progress.

- **Name, Design and Grammar**: Your team should submit a **PDF document** that contains the name, design and the grammar of the language driving the implementation of the compiler and parse tree interpreter/runtime environment. Include information about the interpreter. Discuss the parsing technique you will employ. Also discuss any data structures used by the parser and interpreter. Your team must also submit the **contribution.txt** from your repository's doc folder detailing the individual contribution of each team member until now and the contribution plan for the final milestone.

- **Milestone 2[Due: 4/28 (Hard Deadline)]**: **Final Project Submission**: Final project code, PDF copy of your presentation and YouTube video demonstration of your final product (minimum 10 minutes and maximum 15 minutes length of video). The YouTube video demonstration should walk through the full design and implementation of the language, including the grammar, lexical analyzer, parser, runtime, installation and execution of at least 5 sample programs covering all the constraints noted above. Your team must submit the **following on canvas**:
  - Final Project Code (downloaded from GitHub) packaged in a zip file that includes the updated contribution.txt file.
  - Link to GitHub repository (Please ensure it is shared with the Instructor/Graders with read only access)
  - Link to YouTube video

Before Submission

- Check to assure your solution conforms to the directions on this page.
- Check grading criteria for this assignment to be sure you address all criteria and constraints defined on the assignment page.
- Clean your solution directory of any temporary files. Pull your solution from the remote (GitHub) site into another directory, build and execute the solution to ensure everything has been pushed