CIS Assignment 7 Project Report

Group Members:

Isaac Baah

William Yznaga

Photo IDs:



# Work Per Team Member

## Majority work areas per team member

### Isaac:

* Design of outer loops and design of top-level program procedural flow
* Evaluation of top-level procedure flow for feasibility
* Comprehension of original assignment specification
* Design of scanning/parse tree top-level organization and tree data structure
  + Namely, the TreeNode class
* Debugging output checking

### William:

* Design of inner loops/”concrete”/object-level input-line-scanning and parsing algorithms
  + Namely, the static method parse()
* Implementation of main() method and constructors for TreeNode and Main classes
* Implementation of static compile() method
* Comprehension of top-down/left-recursive compilation/grammar substitution model
* Debugging input checking and overall administration of debugging

# Code/Modules

* Code is written in Java, compiled according to the Java 8/Java 1.8 Standard Edition
* Developed and tested in Eclipse
  + With collaboration supported by usage of a GitHub repo created by W. Yznaga.
* In this Java program, the main operations are:
  + top-level iteration/for loops which:
    - open read access to the input file (with suffix “.c”; tested in this assignment only on input “sample.c”) and get each line to be stored into an iterable data structure;
    - call the static `parse ( )` method on each line of the input (for each line is assumed to be a statement in a statement-list);
    - call the static `compile ( )` method on each statement.
* Main modules of code itself are:
  + main ( ) method
    - executes the above top-level loops and instantiates auxiliary TreeNode object(s)
  + parse ( ) method (**description continues on next page**)
    - Given a TreeNode object (and at root, an individual statement-type node) and reference to a Main object:
      * Can build a **correct** parse tree from any top-level <stmt> and downward, descending top-down and left-recursively
    - Method mainly makes heavy use of tail recursion and switch-statement-case-block control flows to conditionally build parse subtrees from encountered nodes
  + compile ( ) method
    - Given a TreeNode object (and at root, the same kind of node as is given to `parse ( )` at root) and reference to Main object:
      * Can compile an **incorrect at this time** linear series of procedural SIC/XE instructions isomorphic in flat structure to the tree structure of the (*correct*) parse tree
    - Method also, mainly makes heavy use of tail recursion and switch-statement-case-block control flows
      * But in this case, to conditionally generate (sometimes-correct, sometimes-incorrect) SIC/XE source from any given parse-tree node

# Data Structures

* Mainly uses ArrayLists
  + Of <Object>s at base
  + Of <String>s
    - In the case of all grammar terminals except `intnum`s
  + And of mixtures which at root are of <Object>s but the first part of which are <String>s and the second part of which are <Integer>s
    - In the case of `intnum`s alone, for the latter case
* Also uses a single passed-by-reference instantiation of the `Main` class itself
  + To provide `Main.new TreeNode()` instantiation of `TreeNode`s
* Uses the inner class `TreeNode` itself for creating trees and subtrees and their nodes for parsing and compilation purposes

# Bugs and Runtime Status

* After several tests using the `sample.c` output as compared with the `sample.asm` output from the assignment specification document, there are currently:
  + **no** known bugs in the opening input files for reading nor in reading them line-wise
  + **no** known bugs in the parsing and parse-tree generation
  + bugs of **incomplete compilation** of <expr> and <term> nodes
    - in which sub-elements/subtrees are **not** added, subtracted, multiplied, or divided correctly in parentheses-associatie groups
  + **no** known bugs in the compilation of `id` or `<intnum>` grammar-terminal nodes
  + **no** known bugs in root-node-level compilation of <stmt>s into storage-into–`id`-variables
  + bugs of **incorrect compilation** of `RESW` directives as appended to the end of the SIC/XE (psuedo)code source code output

## In Summary

The program:

* + - *Does*:
      * Correctly open-for-reading and read-in linewise input files, and *does* parse the string data linewise correctly
      * Generates a correct parse tree
      * Create and write to a SIC/XE output file *at all*
      * Correctly compiles only a subset of terminals
        + Namely `id` and `intnum` nodes
    - Does *not*:
      * Encounter any “javac”-invocation compilation errors
      * Encounter any runtime errors or crashes or race conditions
      * Compile correctly or at all any nested–`<expr>`-in–`<factor>` nodes, parent–`<term>` nodes, and top-level parent–`<expr>` nodes