**CASE WESTERN RESERVE UNIVERSITY**
Case School of Engineering
Department of Electrical Engineering and Computer Science
EECS 337 Systems Programming (Compiler Design)
Fall 2010
Due: December 2, 2010
100 points

## Introduction

A language compiler and linker normally translate a source program into an executable image for a target computer. Another operation of a compiler is to translate one programming language into another. This type of compiler is called a cross compiler. For this project you shall write a cross compiler to translate Java source code into C or C++ source code.

## Requirements

The name of the compiler shall be java2cpp or java2c and shall run as a console application under Linux or Unix. It shall input a Java source program (Program.java) and output either a C or C++ source program (Program.c or Program.cpp).

Each students shall design and implement a version of the cross compiler. At the end of the project, each student shall submit their final source code to the Digital Drop Box under Blackboard.case.edu.

## Instructions

From a console window, make a directory on your computer in your EECS337 directory under your Case ID and call it project.
**mkdir ~/EECS337/caseid/project/** ; where caseid is YOUR Case ID, in lower case

Change directory to the project directory.
**cd ~/EECS337/caseid/project/**

Download a copy of project_caseid.tar file to the project directory from the EECS337 homework assignment area. To untar the tar file type the command:
**tar xvf project_caseid.tar**

The following files will be created in the current working directory.
java-grammar.zip
project_test.sh
main.c
java11.y
java.l
yystype.h
Makefile
Program.java
course_layout.txt

The original files have been added to include a main program (main.c), an include file (yystype.h) and a Makefile. The java11.y and java.l files have been modified to include a minimum set of routines to build the skeleton java2cpp compiler. The java2cpp compiler has a scanner (java.l) and parser (java11.y) to handle the java programming language. This is the same starting point as the ansi_c compiler from assignment 02. The java2cpp compiler also does not have any actions in the scanner or parser to generate any output.

Notice: The java-grammar.zip file contains the original version of the java grammar (java11.y and java.l). To unzip the file type the following command:
**unzip java-grammar.zip**

A java-grammar/ directory will be created with the original java11.y and java.l files. To look at the differences between the original and modified versions type the following commands:
**diff java.l java-grammar/**
**diff java11.y java-grammar/**

To build the initial java2cpp compiler type the commands:
**make clean**
**make**

Decide if you want to start with the original versions of the java grammar or use the modified files already include from the tar file. Then decide if you want to implement a Java to C or Java to C++ compiler. If you decide to implement the Java to C compiler then edit the Makefile and main.c files. Replace all java2cpp with java2c. You are now ready to implement the compiler project.

**Implementation Details**

Implement the actions to perform the cross compiler operations. Write your own or reuse any of the code from the assignments. Add functions to the scanner to include passing the attributes to the parser. Use dynamic memory allocation routines (malloc and free) to encode the attributes. Use the parser productions to link the memory attributes together and translate the Java productions into C++ or C code. Pass the final linked list to the code generator. The code generator shall output the target source code to a file with the same input file name, except change the file extension from .java to .cpp or .c. Example: java2cpp < Program.java > Program.cpp or Program.c

A sample Java program is provided Program.java. This Java program is written using only procedural methods so can be translated to either C or C++. Additional samples shall be provided on Black board. Once your compiler successfully translates Java source files into a valid C++ or C files then your project is complete.

**Submit Results**

Submit your project source code to the EECS337 digital drop box under http://blackboard.case.edu/. Create a tar file from your ~/EECS337/ directory using the following command and using your case id.

**cd ~/EECS337**
**tar cvf project_caseid.tar caseid/project/***

This command will create a project_caseid.tar file using the files from your caseid/project/ directory. Notice it is important to include the directory path as part of the tar file create. This allows for each students source code to be recreated under a directory using your caseid. This tar ball shall contain the final version of your java2cpp compiler for the EECS337 Compilers class.

Upload your tar file (project_caseid.tar) to the digital drop box under the EECS337 class. Your final directory structure for the java2cpp compiler should be as below:
EECS337/caseid/project/java2cpp or java2c
EECS337/caseid/project/java11.y
EECS337/caseid/project/project_test.sh
EECS337/caseid/project/main.c
EECS337/caseid/project/Makefile
EECS337/caseid/project/java.l
EECS337/caseid/project/yystype.h