Compilers [Fall 2015]

Programming Assignment I

Objectives

1. To understand the definitions of DJ and DISM, which will serve as source and target languages for a compiler built in future assignments.
2. To implement small DJ and DISM programs.
3. To become familiar with and able to use a DISM simulator.

Due Date: Sunday, September 6, 2015 (at 11:59pm).

Machine Details

Complete this assignment by yourself on the following CSEE network computers: c4lab01, c4lab02, ..., c4lab20. These machines are physically located in ENB 220. Do not use any server machines like grad, babbage, sunblast, etc. You can connect to the C4 machines from home using SSH. (Example: Host name: *c4lab01.csee.usf.edu* Login ID and Password: <your NetID username and password>) You are responsible for ensuring that your programs compile and execute properly on these machines.

Assignment Description

For this assignment, you will acquaint yourself with the DJ and DISM languages by implementing one small program in each language. You will write a DJ program in a file called *edgy.dj* and a DISM program in a file called *edgy.dism*.

The desired functionality for your programs is to input a sequence of directed edges in a 100-node graph and then output all the nodes reachable in one step from given source nodes. Nodes are numbered from 1 to 100. The user enters a sequence of edges as pairs of natural numbers; the first number in each pair denotes the source node, and the second number denotes the destination node. Self-loops are allowed, so for example, (4,4) is a valid edge. The user indicates that all edges have been input by entering a 0 as a source or destination node. After entering that first 0, the user may enter a sequence of node numbers; for each such number *n*, a list of all the nodes reachable in one step from node *n* must be output, with the nodes output in ascending order without repeating numbers. The user may terminate this entry of a sequence of node numbers by entering a second 0. The user may also terminate execution at any time by entering a number greater than 100.

The following page shows four examples of this desired functionality. Each example is a trace of running the program in *sim-dism*, so each example ends with a statement indicating that the simulation has completed.

Examples of Desired Behavior:

```
Enter a natural number: 4
Enter a natural number: 4
Enter a natural number: 3
Enter a natural number: 4
Enter a natural number: 3
Enter a natural number: 3
Enter a natural number: 2
Enter a natural number: 4
Enter a natural number: 2
Enter a natural number: 3
Enter a natural number: 2
Enter a natural number: 2
Enter a natural number: 1
Enter a natural number: 4
Enter a natural number: 1
Enter a natural number: 3
Enter a natural number: 1
Enter a natural number: 2
Enter a natural number: 1
Enter a natural number: 1
Enter a natural number: 0
Enter a natural number: 1
1
2
3
4
Enter a natural number: 2
2
3
4
Enter a natural number: 3
3
4
Enter a natural number: 4
4
Enter a natural number: 0
Simulation completed with code 0 at PC=24.

Enter a natural number: 0
Enter a natural number: 1
Enter a natural number: 0
Simulation completed with code 0 at PC=24.

Enter a natural number: 1
Enter a natural number: 0
Enter a natural number: 1
Enter a natural number: 0
Simulation completed with code 0 at PC=24.

Enter a natural number: 4
Enter a natural number: 101
Simulation completed with code 0 at PC=24.
```

Hints
Whenever a DJ or DISM program attempts to read a natural number, the prompts of "Enter a natural number: " get printed automatically. Hence, you don't need to worry about outputting those prompts. DJ and DISM programs can only input and output natural numbers (using the *readNat* and *printNat* calls in DJ, and the *rdn* and *ptn* instructions in DISM).

DJ's assert expression can be used to terminate execution in arbitrary locations.

To give you a rough idea of the effort involved: It took me about 3 hours to do this assignment, in which I wrote 70 lines of DJ and 25 lines of DISM (not counting whitespace/comments). If you find yourself spending significantly more time (like over 12 hours), please consider asking the TA for help with whatever is slowing you down.

Testing Your DISM Program
Please use the DISM simulator, sim-dism, to test your DISM program. When your DISM program halts, it may halt with any code.

Testing Your DJ Program
Because you're writing a program in a new language for which no compiler yet exists, you can't test your program by executing it! This situation is unpleasant but realistic. You'll have to ensure by hand that your DJ program is valid and would behave correctly if executed.

Although you can't directly test your DJ program, you can test it indirectly by modifying it (by hand) into an equivalent, valid Java program. You can then compile and execute that Java program. If you want to test your *edgy.dj* in this way, you may find Java's Scanner class helpful for mimicking the behavior of DJ's *readNat* expression. Documentation is at: http://docs.oracle.com/javase/6/docs/api/java/util/Scanner.html

Extra Credit
To help populate your test suite of programs, you may, for up to +10% extra credit, write and submit two additional programs, *reach.dj* and *reach.dism*. These programs must behave exactly like *edgy.dj* and *edgy.dism*, except that instead of outputting all the nodes reachable in one step, they output all the nodes reachable in any number of steps.

Formatting, Grading, and Submission Notes
- To make it easier for our teaching assistant to read and evaluate your code, use spaces rather than tabs in your code and avoid long lines of code (I try to limit lines to 80 characters in width).
- Your programs will be graded on both correctness and style, so include good comments, well-chosen variable names, etc. in your programs. For full credit, your code must not be significantly more complicated than necessary.
- The teaching assistant will test submissions on inputs not shown in the examples above.

- Type the following pledge as an initial comment in every file you submit for this course: "I pledge my Honor that I have not cheated, and will not cheat, on this assignment." Type your name after the pledge. Not including this pledge in a submitted file will lower your assignment grade 50%.
- Upload and submit your files for Assignment 1 in Canvas. Please upload and submit each file by itself; do not zip them into one.
- You may submit your assignment in Canvas as many times as you like; we will grade your latest submission.
- For every day that your assignment is late (up to 3 days), your grade reduces 10%.