Project #3

Introduction to Bison: McCabe Cyclomatic Complexity

CpSc 8270: Language Translation

Computer Science Division, Clemson University

Brian Malloy, PhD

September 25, 2018

Due Date:

In order to receive credit for this assignment, your solution must meet the requirements specified in this document and be submitted, using the handin facility, by 8 AM, Friday, October 19^{th}, 2018. The handin close date is set at three days after the due date. If you submit after the due date but before the handin close date there will be a ten point deduction. No submissions will be accepted after the handin close date and no submissions will be accepted by email.

How to Submit:

To submit your solution, fill in the README file in the project directory with your information. Then make clean on the project directory, compress the project directory using tar or zip, and submit the compressed directory using the handin command.

Project Specification:

The purpose of this project is to help you to become familiar with (1) Python 2.7, (2) with Bison, and (3) with the Python grammar. The project directory contains a subdirectory, mypy, which contains a scanner and parser generator for Python 2.7; using the files in mypy you can parse virtually any Python 2.7 program and you can verify this by running the file alltest.py in the mypy directory. By starting with a valid parser it will be easier to write test cases to test your metric calculation.

To complete this project you must insert user code into parse.y, scan.l, and main.cpp, to enable computation of McCabe's metric for each function in a Python program, including constructs that were back ported from Python 3x into Python 2.7.2. Of course you can add classes and functions as needed – this is your project, your code. The oracle for the project is the radon code metrics tool, which you can use to compute McCabe's metric. To receive full credit for this project, the output from your metric tool must precisely match the output of the radon tool, which will enable you to use the test script, test.py, in the mypy directory. In addition, you must write enough test cases in the cases directory to adequately test all of the Python constructs that effect McCabe's metric calculation. If the output of your tool matches the output of the radon tool then your test case will pass; otherwise the test case will fail.

The radon tool is fairly well documented and easy to install:

```
https://radon.readthedocs.io/en/latest/intro.html#
https://radon.readthedocs.io/en/latest/commandline.html

sudo apt-get install python-pip
pip install radon
radon cc -s main.py
```

Not All Equal:

Cyclomatic complexity tools will typically report different results and sometimes these results are drastically different. To compare complexity results you might consider installing the flake mccabe module as a basis of comparison: You can install the flake mccabe module using one of the following commands:

```
$ pip install mccabe
$ pip install --user mccabe
```

And you can execute the module on a Python program called main.py as follows:

```
$ python -m mccabe main.py
$ python -m mccabe --min 5 main.py
```

Documentation for the flake mccabe tool can be found at:

`https://pypi.python.org/pypi/mccabe`

Glossary of Files:

- alltest.py – a test script that will execute the test cases one directory away, copied from the Python development web page, with 1537 test cases Passed, and 3 test cases Failed.
- test.py – a test script that will compile your program if necessary, run the test cases in the directory cases, and compare the results from your program with the results of the radon tool.
- scan.l – specs for scanner generator for Python 2.7
- parse.y – specs for parser generator for Python 2.7.
- README – template for you to supply information about you and your project.