Assignment xyz, TCSS 480 Winter 2017
Due Friday, Mar. 3, 2017, noon

OBJECTIVE

The objective of this assignment is to apply your newly learned Erlang knowledge to build a slightly larger application.

ASSIGNMENT DESCRIPTION

Given the initial text file with arithmetic expressions containing (+-/*), you are to create an evaluator program. You need to generate a text file containing expression results.

Input: a txt file named *expressions.txt* (hardcode the file name) with several arithmetic expressions (one expression per line). The operands you need to handle consist of integers and variables *a* and *b*, as defined in *expr.erl*. Sample input file provided.

Expression evaluator: The original file *expr.erl* has been given to you and explained in the series of videos you were to watch over the last couple of weeks as a part of your homework. You will need to watch one remaining video in the series: Video 5 – compiling and running on a virtual machine (https://www.youtube.com/watch?v=ZfC7fjgh5hI&index=5&list=PLR812eVbehlwEArT3Bv3UfcM9wR3AEZb5) , and then extend this program with the following:

- add subtraction and division operations to both eval and compile-execute methods
- handle blanks in string expressions so that one can type in an expression containing space key character anywhere in an expression, e.g. "(2 * (3 + 4))" instead of "(2*(3+4))"
- add operator precedence so that parenthesis may be omitted, e.g. "(2 + 4 * 3)" is a valid expression equivalent to "(2 + (4 * 3))"
- add conditional if – then – else to the evaluator so that if the "if" expression returns a non-zero number, then "then" value is returned; otherwise, "else" value is returned,
  e.g. "if (2 + 2 – 4) then 4 else (2 * 3 + 5 * 2)" returns 16 whereas
  "if (2 + 1 – 5) then 4 else (2 * 3 + 5 * 2)" returns 4

Concurrency: It is up to you to decide how to implement concurrency in this assignment. One of the possibilities is to spawn one process per each line to be evaluated and/or spawn one processes that evaluates an expression using eval and another using compiler-execute.

Test file: You need to provide a test file that will test the following functions:

- parse
- eval
- execute

Other specs: You need to name your program *assign9.erl* and your test file *assign9_tests.erl*. *assign9.erl* should contain functions called *mymain1()* and *mymain2()*. The first one is to run the program on *expressions.txt* using *eval* function and is to write the output to an output file. The second one is to run the program on *expressions.txt* using *execute* function and is to write the output to another output file.

Executive summary

You are to also write an executive summary similar to the ones you used to write in TCSS 305. It should be written as a txt file. In the file, describe what you have done to complete the assignment in 200-500 words. The word count is not strict, so do not worry about going slightly over; however, summaries that do not meet the minimum length requirement or are trivial in nature (representing little thought or effort) will not get full credit. You can share your personal experiences, things that particularly frustrated you about the assignment, things that particularly interested you about the assignment, etc. It is especially important that you document any difficulties you had with Erlang, the libraries, etc.

SUBMISSION AND GRADING NOTES

You need to submit *assign9.erl* and *assign9_tests.erl* through *Canvas*. Your code needs to include comments and use consistent coding style – include the header with your name, program name, date; function headers that describe the purpose of each function; code comments that describe the high level logic

This is an individual assignment – you are NOT allowed to work on it with any other student or share your code with others. This assignment is NOT subject to a peer review process and will be graded by the instructor or a grader.

The code will be graded by running it on different versions of *expressions.txt* and by running your own test cases.

Grading points will be distributed in the following fashion:

- Test file showing the proper processing of 25 pts
    - parse
    - eval
    - execute
- Program functioning on various *expression.txt* files 12.5 pts
- Parallelization 7.5 pts
- Coding style and comments 5 pts