Dear students,

Below are the most frequently asked questions of this week's lab (week 5):

Question 1: What's the difference between the Tarantula score and the fitness of an individual?

Answer:
The Tarantula score is a metric that is used to localise the possible faults in your program and should not be used as fitness. The fitness (e.g. proportion of tests passed) of an individual is an indication of how well an individual serves as a solution for a given problem. In our case, we want to find an individual that can pass all tests. This individual would then be a candidate patch for the buggy problem. It is also possible to compute the number of faulty operators that your genetic algorithm has fixed as the fitness of your individual. In this case, you want to find an individual that fixes all the buggy operators.

Question 2: Are you supposed to do anything in the "`encounteredOperator`" method?

Answer:
Yes, this method is called whenever a binary operation is called in the original problem. If you read carefully, the result of the operation is based on the operator that is used (because of this line: `String replacement = OperatorTracker.operators[operator_nr];)` You would want the replacement to be an operator from the individual that you are evaluating. You have the operator number at your disposal (the number is the same order in which the operators occur in the original problem file), so you can use this to get the right operator for the right operations. In this method, you can also keep track of which operators were used for a given test case (using the `current_test` attribute).

Question 3: I am getting a list of only "true" after running "`runAllTests`", am I doing something wrong?

Answer:
You will get a list containing only "true" back from "`runAllTests`" if you are using the original RERS problem instead of the buggy RERS problem. There are two things which you can do to see whether it fixes your issue:

1. Check whether you have run "instrument_buggy" on VS code. Make sure that you selected "patching" for the type of instrumentation. Also, make sure that you have put the "RERS2020Buggy" folder and its contents (ZIP is provided to you on BrightSpace) in the root directory of the JavaInstrumentation project.

If you are not using VS Code, make sure that you have instrumented buggy RERS problems that are provided to you on BrightSpace using the “patching” option.

2. Check whether there is only “true” in the list returned back by “`runAllTests`”. There are many test cases written for each buggy RERS problem and we randomly flipped a few operators in the original RERS problem to introduce bugs. Naturally, you will see much more “true” than “false” in the test results; the passing tests will drown out the few failing tests.

Question 4: Should we consider each operator as an individual for our GA or should we use the whole operator list?

Answer:

You should use the whole operator list as your individual; the whole operator list should be seen as the “gene” of your individual. In the lecture, an example was given of generating test cases for a program. In this example, all test cases written for a class/component are considered as the gene of your individual and you do a crossover between two lists of test cases to generate a new list of test cases (child).

Additionally, if you are considering an operator to be an individual in your GA, then it would not be possible to do crossover anymore; you cannot split an operator in half and use it to create a new individual (child).

Question 5: When we do a crossover, we will end up with fewer individuals in our new population, won’t our population keep growing smaller after each run of our GA?

Answer:

Yes, your population will indeed become smaller if you only use the children that are created from crossovers as individuals for your new population. You can do many things to fill you population back to the initial size that you started with:

- The tarantula score tells you which operators are suspicious and you might want to replace these operators. You can create different sets of replacements (replacing only one, replacing two, replacing three etc.) and each can be used to create a new individual. You can use this method to fill up your population.
- You can also add a few individuals that have completely random operators. This might help with jumping out of local minima/optima

- You can add the parents that were used for crossover back into the population (their genes also survive another round). This does assume that you only do crossovers between parents that have high fitness.