# Grammars and Testing

Task 1 – Mini lecture

Task 2 – Share your solution to the directory reader lab with your teammates. Note differences and similarities in your solutions. Do you think that there is a non recursive way to solve this problem? What are the advantages in using a recursive solution here?

Task 3 – Exploring the grammar.

Grammars are ways of describing constructs. They establish the syntactical rules for a language.

Given the example grammar (similar to your PA), answer the following questions:

1. Which of the following are legal statements in the grammar? For each correct expression, show what it evaluates to.

a. 3

b. 3 + 4

c. 3 + 4 – 5

d. (3 + 4)

e. (3)

f. 3 \* 4 + 2

g. 3 \* (4 + 2)

h. ((3) + 4

i. –(3 \* 4)

j. 2 + 3 / 4 \* 5

2. For the following expression, show how it should be parsed into its various terms. In other words, if it is an expression, it is made up of terms, factors, and expressions. Show how you would use those constructs to parse this expression.

3 + 4 \* 5 / (7 – 2) \* 17

#### Task 4 Introduction to testing

Testing is a process that every program must go through to verify that it works properly. Today we will focus on Black box testing…testing the program based on input and expected output.

For PA4 (a copy of which you have for your group), design test cases to thoroughly test the program. You do not need to test the same things, but should design each test looking for something different. You should test the “normal” cases, boundary cases, 0 cases (or empty cases), and any error condition that is described. (In a more critical system one would also test error conditions that may not be directly described.) Describe your tests on the chart on the back of this page. When your group is done, put two of your test cases on the board, not duplicating the test cases of any other team. (In other words we should have unique test cases on the board. NOTE: The number of rows is not intended as a hint about the number of test cases you should have.

|  |  |
| --- | --- |
| **Description of case** | **Description of behavior** |
| EXAMPLE:Call the program without passing in a command line argument. | Program should display the message, “Missing file name” and exit. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |