Assignment 4 – Expression Parser
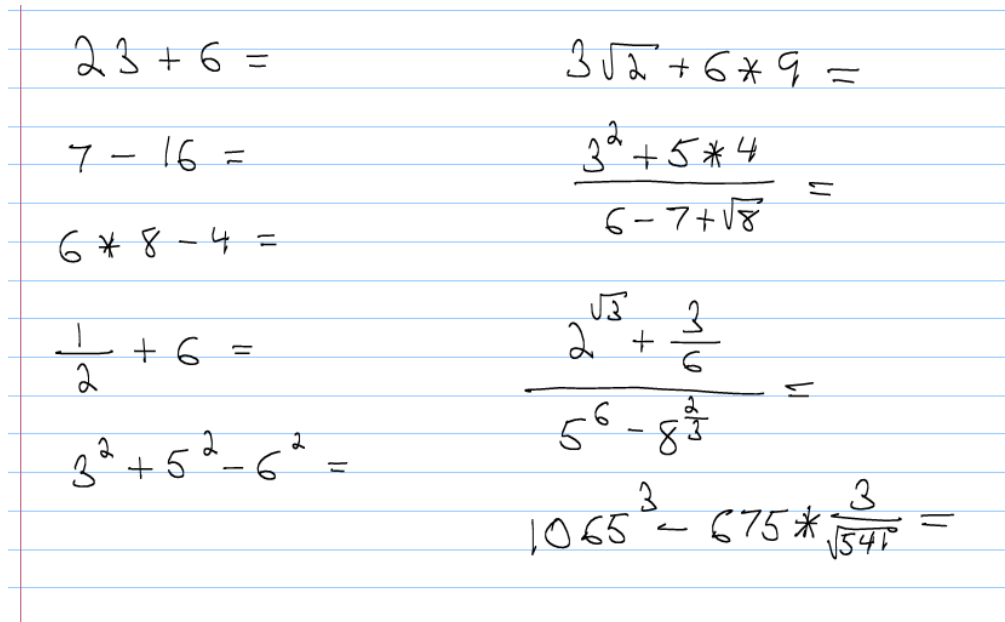## CAP6105

Due: 11/4/16 11:59pm

The focus of this fourth assignment is to learn the intricacies of creating a 2D parsing system for recognizing mathematical expressions. This is the second part of a two part assignment where you will be creating a simple pen-based calculator. This assignment requires you to take recognized symbols and understand the spatial relationships between them. This understanding will let you create parse trees for mathematical expressions that can be evaluated using simple tree traversal.

Requirements

Your expression parser must be able to take a list of recognized symbols and their bounding boxes and output a parse tree so the expressions you write can be evaluated.

Your parse should support addition, subtraction, multiplication, fractions, superscripts (one level only), square roots, and the equal sign.

Examples of some expressions your expression recognizer should be able to handle include:

$$23 + 6 =$$

$$7 - 16 =$$

$$6 * 8 - 4 =$$

$$\frac{1}{2} + 6 =$$

$$3^2 + 5^2 - 6^2 =$$

$$3\sqrt{2} + 6 * 9 =$$

$$\frac{3^2 + 5 * 4}{6 - 7 + \sqrt{8}} =$$

$$\frac{2^{\sqrt{3}} + \frac{3}{6}}{5^6 - 8^{\frac{2}{3}}} =$$

$$1065^3 - 675 * \frac{3}{\sqrt{541}} =$$

You should be able to invoke the expression recognizer using a simple equal sign and tap gesture and you should also be able to use your scribble gesture to erase symbols. In addition, you should be able to move mathematical symbols around using your finger.

For extra credit, you can support the trig functions tan, cos, and sin. Note that in order to use these functions you will need to add parentheses to your symbol recognizer.

Strategy

You can utilize any of the parsing strategies we discussed in class. However, I would recommend you use a 2D coordinate grammar since this is, by far the easiest to design and implement. If you use this approach, your parsing routine will take, as input, a list of recognized strokes along with their bounding boxes. I would also write down a simple grammar for your parsing engine. This will help you get started in thinking about the problem and how to design your algorithm. I would also recommend using a parse/process strategy where you have a set of parse functions that act on your grammar and a set of process functions that perform spatial relationship tests.

Once you have a parse tree for a given expression, it is straightforward to perform an evaluation. You simply have to traverse the tree and use a switch statement on the operators. Note you do not need to write a piece of code to evaluate expressions that must be compiled and run in real time. You can simply have a compute function that gets called when you want to evaluate an expression.

Things you should keep in mind.

1. You once again can run in real time or in batch mode (for ex. lassoing the symbol or symbols and taping to invoke the recognizer).

2. For evaluating expressions you can perform the operation right after recognition, or with a specific command. I have always preferred the equal/tap approach since is so intuitive from the user's perspective.

3. You will need to show recognition results to the user. A simple text box is fine but if you want to be more elaborate feel free to do so. You will also show the output of the any expression evaluations. You can put the result to the right of the equal sign.

Deliverables

You must submit a zip file containing your source and any relevant files needed to compile and run your application to the ISUE lab drive Also email me a README file describing what works and what does not in your application, any known bugs, and any problems you encountered. Please include a file I can open in your application that has the mathematical expressions you used to test the recognizer written down in ink.

Grading

Grading will be loosely based on the following:

80% correct functionality
20% documentation