1) Describe any challenges you encountered

I spent lots of time thinking how to make the trees be minimum. I spent two days building my program ( with help of the code<https://www.geeksforgeeks.org/expression-tree/> I found online, which helped me think of the structure of my code more easily). Since the code on the site only includes the postfix program and without the minimization, I spent two to three days on adjusting my code and making it available to read input and do the prefix and in-fix conversion.

The most difficult part of my program is the commutative law and the distributive law. If I just normally print out the nodes of the tree, what I get would only be the operator and operand without order of priority (parenthesis), hence, I drew trees and imagined the way how nodes stored in the stack, by imaging how the output would look like and count the stack’s node, I finally executed the right program to convert my equation. It is really a great fulfillment to me, no matter what input I enter, this code would convert them into the minimum size.

Also, the condition to judge whether it is time to push and pop in order to build a new tree is complicated to me, if there is some condition I forgot to set, then there might be no output. Scenario error is the most difficult part to deal with.

It is really a time consuming work to me, noticing the fact that this project only stands for 5% of my final grade, but I think what I did and this project is worth 20% of my final grade … . Although it is a little unfair and I think I spent too much time on this project ( I fortunately don’t have midterms of other subjects this week so I got more time to do this project ), I actually found it fun when thinking about how to sort and minimize the equation. This is my first time thinking that I completely completed the program. Is it possible to get extra points if I do well on this work? I didn’t do well on my midterm exam ….

2) What did you learn from this programming assignment?

I learned much about trees and how to use them to store and adjust the data. It is really a good tool to compile the equation. By storing each node, creating, popping and pushing the values of each node, it is very easy to access the wanted data and rebuild a tree.

3) Which data structures did you use, why?

I use stack and tree traversals to execute. By storing each node into the stack and popping them out with the procedure of LIFO, it is easier to pull the value out, adjust it by storing them into the children of nodes and then push them back to the stack waiting for another adjustment.

4) What is the time complexity of your program in terms of comparisons?

The function of comparisons:

post\_fixConvertion: 123nt+t

pre\_fixConvertion: 123nt+t

in\_fixConvertion: 193nt+t

5) Which section(s) of code requires the greatest number of comparisons?

in\_fixConvertion: 193nt+t

6) Why are parentheses not required for pre-fix and post-fix expressions?

The closer the operator is to the operator means the priority of conduction. So, for prefix, the operator that is the closest to the first two operand (beforehand) would be conducted first, regardless of the characteristic of arithmetic; same as for postfix, the operator that is the closest to the first two operand (afterhand) would be conducted first, regardless of the characteristic of arithmetic. Thatis, the order of the operand means the parentheses of infix expression.