Maayez Imam 915342727

report.pdf

Postfix Eval:

With my postfix_eval.cc code, the logic behind it is this: the user inputs a string, so I convert that string to a stringstream so that it parses the characters in the string. This allows me to access each character individually. When I can do that, I can iterate through each string to see if it is a number or an operation, and act accordingly. In order to iterate through each string, I push each string into a vector. I then iterate through that vector and check if the string is an operator or not. If it is, I pop two numbers from the stack and operate on them. I then convert the answer received back into a string and push it back onto the stack. If it is not an operator, I just push it onto the stack. After all the tokens are iterated through, I pop the final result from the stack and print it out as the answer. I keep doing all of this as long as the user still wants to input.

Luggage Handling:

With my luggage_handling.cc code, the logic behind it is this: I read the file and put all the luggage numbers into a vector. Then I push all the luggage (numbers) onto a container (queue). When the container hits the maximum container size (given by the third command line argument), I push the entire luggage container off the plane (onto a stack). But right before that, I reverse the order of the container by pushing them onto a vector and reversing them so that the luggage is stored in the order that they came off the plane. If there is any remainder luggage, I push that out of the plane as well. I then print out the luggage in order of how they got off the plane.