Raymond Chan - ID#:914817541

Program #1: Postfix expression evaluation

The program itself will ask for user input until there are no input detected anymore. Each user inputs are taken by using std::getline, and be stored into a string. Once we get the string, we could tokenize each element from user input by using stringstream.

Each elements should be categorized into two types: double or character. As double, the element will be immediately push into the Stack class, while elements as character need to be verify if the character is one of the four operators: '+', '-', '*', '/'. If the character is not one of the four operators, then we will print out the error statement for unknown symbol, and break the while loop to end the execution of this input. Also, if the Stack itself is empty while a character is read, this signalify that user input is invalid.

Notice that the string "piping" needs to add a space at the end. That is to make sure the last element can be read. Also notice that I also checked if the substring is empty, which skips all the unnecessary whitespaces between the elements.

After finishing the calculation, the program should check if the calculation is truly valid, and print out the answer if it is. But if the final step doesn't meet certain criteria, the program will print error statement and end the execution of this input. Say, if the Stack does not have exactly one element left, that means the user's input does not meet the proper expression structure, therefore the invalid expression should be printed.

The return value of this program, upon successful execution, should be 0.

Program #2: Air luggage management

This program will require the implementation of both Stack and Queue. Since this program needs to read from the file, we will need to check if the input syntax is valid, as well as making sure the file_name is valid also. Each elements from the file will be stored within the vector BAG_LIST, although it is not necessary.

The program will first tries to put elements from BAG_LIST the Queue container, where the Queue is in the form of a bounded array. The Queue will then be pushed into the Stack as a shared pointer (since using unique_pointer require changes to Stack implementation, which I am not going to do so)

Finally, once all elements were processed, it is time to outputting them. The elements will be outputted by popping out the top of Stack first, then popping out the earliest inserted elements inside the each Queue. This process should continue until Queues are empty, and the Stack is empty.

Notice that the return value should be 1 if the program ends with error statements, and 0 if the program were successfully executed.