2) Report

a)  brief description of notable obstacles you overcame.

I have trouble figuring out the code to make sure that it doesn’t get out of bounds and all other possible error scenarios. I also couldn’t load my file into putty for some reason, which made testing hard.

1. A description of the design of your program. You should use [pseudocode](https://web.cs.ucla.edu/classes/spring20/cs31/pseudocode.html) in this description where it clarifies the presentation.

bool isValidUppercaseStateCode(string stateCode)

list out all valid up statecodes

return (string.size() ==2 && does not find ‘.’ && match found)

int CountOrders(string digitsinStr) {

set empty string Orders

loop while i is smaller than orders.size()

loop while i is a digit

add digit to Orders string

increment i

change string into integers

total up the individual state orders

clear up the Orders string

increment i to move on to next position

return totalcases

bool hasValidSyntax(string orders)

set empty statecode string

override function by setting validsyntax = true if orders is empty string

loop while i < orders.size()

while (i is an alphabet)

save capitalized alphabet to string statecode

increment i

if (i == order.size())

break out of loop

if statecode does not match valid statecode, return false

clear the string

if next index is not a digit, return false

while (i is a digit)

increment i

if (i == order.size())

break out of loop

if index at i is not +/-, return false

increment i

returns true if string passes all tests

int countCases(string orders, char status, int& caseCount)

create empty strings for numOrders, PosOrders, NegOrders

if (the string orders does not have ValidSyntax), return 1

while (i is smaller than orders.length)

increase I by 2

while (index is a digit)

add the digit at i to string numOrders

increment i

if any of the cases equals to 0, return 2;

if index followed is +,

settle positive orders

if index followed is -,

settle negative orders

clear numOrders string

loop back up

if status does not equal to +/-, return 3

if status is +

calculate casecount by calling positive orders in CountOrders function

return 0

if status is -

calculate casecount by calling negative orders in CountOrders function

return 0

1. A list of the test data that could be used to thoroughly test your program, along with the reason for each test. You don't have to include the results of the tests. Notice that most of this portion of your report can be written just after reading the requirements in this specification, before you even start designing your program.

* No + or – after number of cases – CA23TX29-
* Invalid state code - ZM23TX29-
* Small letters in code - ca23-TX29-
* Numbers before state code – 23CA+TX29-
* Empty string - “”
* Two different status - CA23+TX90-
* Single digit number of state order – CA3+TX6-
* More than three states- MA54-CA23+TX64-
* Random string – 23ergtj4j6
* Symbol is something other than +/- - MA64+CA8-
* Two consecutive states- CAMA65-
* Space in the beginning- “ CA43-TX5+”
* Space in the middle- CA 43+ TX54-
* Space in the end- “CA43-TX54+ “
* Check if number of cases is reset when return 0 - countCases (“CA43+TX54-“, +, 9999)
* Check if number of cases is not reset when return 1- countCases (“CA@+TX54-“, +, 9999)
* Check if number of cases is not reset when return 2- countCases (“CA0+TX54-“, +, 9999)
* Check if number of cases is not reset when return 3- countCases (“CA45@TX54-“, +, 9999)
* Check only one number is returned when any two of 1/2/3 occurs- countCases (“CA45@TX0-“, +, 9999)
* Check two consecutive zeros in number of cases- countCases (“CA0045+TX6+“, +, 9999)