CS 31 Project 3 Report

1. Brief description of the obstacles you overcame

One obstacle I faced while completing this project was trying to understand how to break down the program so I would not code the whole thing at once. Once example of this was when I was trying to figure out how to write the function to check for the validity of the state code. I had first tried to check for the state code, the numerical value, and the order status all under one function. I later realized that some of the state order parameters required more complex code than others which caused the code under that function to be quite long. In thinking how to simplify the code, I used the isValidUppercaseStateCode function given in the spec to split my code up into multiple functions, and had one call the other which make my program more readable.

Another challenge I encountered was understanding what I was supposed to put in the main function. The spec talked about using assert to test our program, but I still struggled to fully understand this concept. After doing more research to better understand what an assert was, I was able to use this facility to write the tests for my program. I based my code off of the example given in the spec and attempted to cover as many test cases as I could to ensure my program covered the project requirements. Making sure I wrote enough test cases was also a little challenging in itself because I wanted to make sure I did not forget anything. To combat this, I wrote out the project spec requirements at the beginning of my program so that I could better reference them without having to go back and forth between windows.

A third obstacle I had to overcome was making sure I was not too detailed in writing out the pseudocode for the next part of this report. I had to go over my pseudocode multiple times to check whether or not I could better summarize the code rather than restating the code. I had other peers look over my pseudocode as well in case I had overlooked anything that was too specific. While writing, I would also try to limit myself to about five words per line in order to write pseudocode that was readable and understandable without rambling on about the details.

1. A description of the design of your program. Use pseudocode

*Test for isValidStateCode:*

*Repeatedly:*

*Change letter digits to uppercase*

*If valid state code*

*Return the two letters*

*Test for valid syntax:*

*If empty string*

*Return true*

*Repeatedly:*

*If statecode is not valid*

*Return false*

*If not a digit character*

*Return false*

*Repeatedly:*

*If not a digit character*

*Return false*

*Otherwise return true*

*Count cases by doing:*

*If orders does not have valid syntax*

*Return 1*

*If order status is not + or –*

*Return 3*

*Repeatedly:*

*If orders has a number*

*Add that value to count*

*If there is a + or a –*

*If there are 0 orders*

*Return 2*

*If +*

*Add value to number of filled orders*

*Otherwise*

*Add value to number of unfilled orders*

*Main:*

*Test different cases*

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.

(CA132+) for valid syntax

(cA132+) for one variant of the state code

(ca132+) for another variant of the state code

(Ca132+) for another variant of the state code

(CA132-) for order status that is not +

(CA) for invalid syntax; not a complete state order

(CA+) for invalid syntax; not a complete state order

(CA132) for invalid syntax; not a complete state order

(CA0-) if order is 0

(TX38- CA132+) for invalid syntax; space between state orders

(CA 132+) for invalid syntax; space in the state order

(CA132 +) for invalid syntax; space in the state order

(CA 132 +) for invalid syntax; multiple spaces in the state order

(CA132&) for invalid syntax; order status is not + or -

(CH132+) for invalid state code

(CAC32+) for invalid state code and digit value

(TX38-CA132+) for correct syntax with two state orders

(“”) for valid syntax

(“ “) for invalid syntax

(TX38-C) for invalid syntax; incomplete state order

These are just a few examples of the types of data inputs to test.