Major obstacles:

* Be mindful of all requirements about the project spec while building the program (cannot assume anything about the input string, it could be NYNYNY00009- etc.)
* Keep tracking of the counting variable, when it has been incremented when it has not
* Logic condition, && or ||
* Decide the edge value for entering a loop and leaving a loop
* Called a variable that’s not visible to my current working scope

Pseudocode:

Function *bool isValidOrder(string orders)*

1. If user inputs an empty string, it passes the test, we stop.
2. Otherwise, start the outer most loop.
3. Start the 1st inner loop.
4. Find alphabetic characters.
5. Cast alphabetic characters to uppercase.
6. Store alphabetic characters in an empty string.
7. If there are no more alphabetic characters, break out of the 1st inner loop.
8. Check if the alphabetic characters string is a valid state code.
9. Set the alphabetic characters string to empty.
10. Start the 2nd inner loop.
11. Find at least one digits.
12. If there are no more digits, break out of the 2nd inner loop.
13. Find the end of a state order.
14. Check if the state order ends correctly.
15. Loop back to step 2.
16. When the outer most loop is done, and reaches here, the string passes the test.

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

1. If it’s an invalidity order string, end the function.
2. If it’s an invalid status, end the function.
3. If it’s an empty order string, end the function.
4. Start the 1st loop.
5. Find nondigital and nonalphabetic character
6. Start an inner loop
7. Find digit and transform them into *number*.
8. If there are no more digit, the inner loop ends.
9. Check if the *number* equals to 0.
10. Reset the *number* to 0.
11. Loop back to step 4.
12. The 1st loop ends after looping over every character in the order string.
13. Reset the *number* to 0.
14. Start the 2nd loop.
15. Find status
16. Start an inner loop
17. Find digits preceding the status
18. Transform all digits preceding the status into a *number.*
19. If there are no more digits, the inner loop ends.
20. Loop back to step 14.
21. The 2nd loop ends after looping over every character in the order string.

Testing data for function *bool hasValidSyntax(string orders)*

\*Highlighted test data make my function misbehave, but I didn’t realize until I tried them out.

|  |  |
| --- | --- |
| Test data | Reasons |
| “” | Testing empty string, it should pass the test |
| “ ” | Space char should not be recognized as an empty string |
| “ ” | More space char should not be recognized as an empty string |
| “Ca11+” | See if lower case letters would be casted to upper case |
| “cA11+” | See if lower case letters would be casted to upper case |
| “ca11+” | See if lower case letters would be casted to upper case |
| “CA11+” | See if upper case letters would remain unchanged |
| “ CA11+” | Space char at the beginning of the order should be invalid |
| “CA 11+” | Space char in the middle of the order string should be invalid |
| “CA11+ ” | Space char at the end of an order string should be invalid |
| “11+” | See how program handles missing state code |
| “CACA11+” | See if more than one correct state code would be recognized as valid |
| “CACACA11+” | See if more than one correct state code would be recognized as valid |
| “TN11+” | Try some state codes other than CA |
| “WV11+” | Try some state codes other than CA |
| “DC11+” | Try some state codes other than CA |
| “FL11+” | Try some state codes other than CA |
| “XT11+” | See if wrong state code will be detected |
| “AA11-” | See if wrong state code will be detected |
| “@#11-” | See if wrong state code will be detected |
| “CA100CA+” | Testing if the program detects that there shouldn’t be anything other than digits in between letters and +/- |
| “CA100%CA+” | Testing if the program detects that there shouldn’t be anything other than digits in between letters and +/- |
| “CA1-” | Only one digit, see if the loop that checks that there shouldn’t be anything other than digits in between letters and +/- works |
| “CA0+” | Zero order is valid, see if it passes the test |
| “CA00000+” | Zero order could be in 00000 form, but it should still pass the test |
| “CA007+” | See if program recognize 007 as valid |
| “CA123456789-” | Try more digits, see if the loop works |
| “CA+” | See how the program handles missing number string |
| “CA12345%” | See if the program checks the end of a state order correctly |
| “CA123-” | See if the program checks the end of a state order correctly |
| “CA123” | Missing + and – char, see if the program checks the end of a state order correctly |
| “ca123-ny0+” | Zero order in one of the state orders, but should still be valid |
| “ca123++ny3+” | Extra + in between two valid state orders, see if it detects error |
| “ca123+$ny3+” | Extra $ in between two valid state orders, see if it detects error |
| “ca10+ny90-” | See if the program works for more than one valid state orders |
| “ca10+ny90-az100+” | See if the program works for three valid state orders |
| “ca10+ny90- az100+” | Test if the program finds out the space in between correct state orders is incorrect behavior |
| “ca10+ny90-az100+MA200-” | See if it works for four valid state orders |
| “ca10+ny90-az100+MA200-AL345+” | See if it works for five valid state orders |
| “ca0+ny00-az000+MA0000-AL00000+” | All zero orders but should still be valid |
| “ca10+ny90-az100+MA200-AL345+dc67-fl8+IL0-WY007+va567+TX38-CA132-Ms6-nY290-UT006-MS8-CA15-” | Try a stupidly long, but valid order string, see if all loops behave well |
| “ca10+ny90-az100+MA200-AL345+dc67-fl8+IL0-WY007va567+TX38-CA132-Ms6-nY290-UT006-MS8-CA15-” | Try a stupidly long, and invalid order string, see if all loops behave well |

Testing data for function *int countCases(string orders, char status, int& caseCount)*

|  |  |
| --- | --- |
| Test data | Reasons |
| “Ca+”, ‘+’, -999 | Invalid order string should return 1 |
| “”, ‘-’, -999 | See if empty string returns 0 and set caseCount to 0 |
| “”, ‘#’, -999 | See if it returns 3 and caseCount unchanged |
| “cA10+Ny12+WV345-”, ‘@’, -999 | Wrong status input, should return 3 |
| “cA10+Ny12+WV345-”, ‘ ’, -999 | Space char, program should return 3 |
| “cA0+Ny12+WV345-”, ‘@’, -999 | One zero order, and a bad status input, see if it compiles and returns only one number |
| “cA0+Ny0+WV345-”, ‘@’, -999 | Two zero orders, and a bad status input, see if it compiles and returns only one number |
| “cA0+Ny0+WV0-”, ‘@’, -999 | Three zero orders, and a bad status input, see if it compiles and returns only one number |
| “cA10+Ny12+WV0-”, ‘+’, -999 | Placing zero order at the beginning of the loop, see if it returns 2 |
| “cA10+Ny0+WV345-”, ‘-’, -999 | Placing zero order in between correct nonzero orders, see if it returns 2 |
| “cA0-Ny12+WV345-”, ‘+’, -999 | Placing zero order at the very end of the loop, see if it returns 2 |
| “cA0-Ny0+WV345-”, ‘+’, -999 | Testing more than one zero orders |
| “cA0-Ny0+WV0-”, ‘+’, -999 | Testing all zero cases |
| “cA00-Ny000+WV0000-”, ‘+’, -999 | Testing if program recognize 00 and 000 and 0000 as all 0 |
| “cA10+”, ‘+’, -999 | See if it returns 0 for just one order, and set caseCount to 10, initially failed |
| “cA10+Ny12+”, ‘+’, -999 | See if it returns 0, and set caseCount to 22,  Passed after debugging |
| “cA10+Ny12+WV345+”, ‘+’, -999 | See if it returns 0, and set caseCount to 367,  Passed after debugging |
| “cA10+Ny123+WV3456+”, ‘+’, -999 | See if large number get calculated correctly |
| “cA10+Ny12+WV345+”, ‘-’, -999 | All cases are in the opposite status, testing if it sets caseCount to 0 |
| “cA10-Ny12-WV345-”, ‘+’, -999 | All cases are in the opposite status, testing if it sets caseCount to 0 |
| “cA10+Ny12-WV345+”, ‘+’, -999 | See if it skips - sign and just adds numbers belong to + |
| “cA10+Ny12-WV345+”, ‘-’, -999 | See if it skips + sign and just adds numbers belong to - |
| “cA10-Ny12+WV345-”, ‘-’, -999 | See if it works for - sign |
| “cA010+”, ‘+’, -999 | See if it recognizes 010 as 10 |
| “cA010+Ny012+”, ‘+’, -999 | See if it recognizes numbers correctly with two orders |
| “cA010+Ny0012+WV000345+”, ‘+’,  -999 | See if it recognizes numbers correctly with three orders |
| “cA0100+”, ‘+’, -999 | See if it recognizes 0100 as 100 |
| “cA0100+Ny0120+”, ‘+’, -999 | See if it works for numbers like 0120 in two orders |
| “cA0100+Ny0120+WA03450+”, ‘+’,  -999 | See if it works for numbers like 0120 in three orders |
| “ca10+ny100-az100+MA100-AL345+dc100-fl8+IL100-WY007+va100-TX38+CA100-Ms6+nY100-UT006+MS100-CA15+”, ‘-’, 999 | Try a stupidly long, complicated but valid order string, see if it returns 0 and set caseCount to 800 |