David Li

404473282

1. In this project, I had several obstacles. I had difficulty trying to make the “isWellFormedString” function and its small exceptions for certain strings such as making sure even 01, 02, 03 etc. does not count as a valid integer for a cookie/pastry count. When I made asserts to test whether my function worked, there were several times when there were some errors. I had trouble making sure that reading two nonalphabetical and nondigital characters such as “+” or “/” or “-“ was considered not well formed. Because my wellformed Boolean value was by default true, I had to find all the exceptions for false. Maybe next time I can try the other way – making the default false and then finding all the exceptions to make it true (depending on whatever is quicker and easier)
2. I created the program with 4 functions. One function is the isWellFormedString function, which reads in each character of the string and determines if it’s an alpha or digit value and from there, determines through certain if statements whether it fits the definition of a valid string or not. Another function was the calculateOrderCost function, which made sure that the isWellFormedString function would return true, then it would run through calculations after counting the number of dessert orders and calculating it via the given project instructions.

One of my other functions was used to convert a char value into an int value for calculation usage. Another function was used to make all the string characters into lower case characters to avoid further complications. Finally, there was a main function used for assert statements.

1. (“”) – test that empty strings are considered invalid (return false) (and return $0 in cost).

(“12r”) – test that invalid character is considered invalid.

(“B”) – test that a single character won’t be counted as “1B” but rather as invalid.

(“33”) – Test that solely an integer is invalid.

(“13B”) – test that a valid digit and a valid character will return a true statement and a correct order cost of $8.25.

(“3c”) – test that a valid digit and a valid character (even lowercase) will return a true statement and a correct cost of $1.25.

(“1c36B1i36d1S”) – test that an order string with valid digits and with all the possible dessert options will return a true statement and a correct cost of $98.50.

(“3cc”) – test that a repeated character will return a false statement.

(“3c39b”) – test that an order with over 3 dozens of one item will be considered invalid.

(“36C1C12B”) – test that an order adds up one item over 3 dozens will be considered invalid.

(“+36c”) – test that a nonalphabet and a nondigit character in an order string will return false.

(“36c 36b”) – test that a space in an order string is considered invalid.

(“12+-“) – test that more than one invalid symbol repeated is considered invalid.

(“+-“) – test that only symbols will be considered invalid.

(“12+-12C”) – test invalid symbols in the middle of an order string will be considered invalid.

(“36C 12 B”) – test that more than one space in a string will return false.

(“37C37B”) – test that an order with two items over 3 dozens will be considered invalid.

(“09C2B”) – test that an order with a “0” in the tens place at the beginning of a string will be considered invalid.

(“9C02B”) – test that an order with a “0” in the tens place in the middle of a string will be considered invalid.

(“1C 1c 2b”) – test multiple spaces in an order string will return false.

(“0c”) – test that an order with 0 dessert orders will return false.