1. Obstacles:

* For **isValidEbayListingString(string auctionstring)**, I encountered several problems when looping through the auctionstring. When the character was ‘+’, my code initially only tested whether the character in the next index was a digit and the character in the previous index was ‘B’ or ‘b’. So, when the for loop checked the case for the next character, there was no case checking what action to take when the character was a digit and the character before it was ‘+’.
* **#include <cctype>** was not included in my initial code, which made checking a lot of the integer characters slightly more tedious.
* Initially, the **getVal(string auctionstring, int & i)** was not working for the whole string and just converted a character to a number. This caused several problems for the functions **listingSold(string auctionstring)** and **howMuch(string auctionstring).**
* For the **watchers(string auctionstring**) function, I had initialized a variable of integer type in the ‘for’ loop, which caused that variable to keep getting reinitialized and did not keep track of the number of watchers accurately.
* For the **howMuch(string auctionstring)** function, for the case when the character is a ‘B’ and the next character is ‘+’, I converted the ‘+’ to a number using the **getVal(string auctionstring, int & i)** function instead of converting the digit after the ‘+’. This caused several miscalculations in **bidPrice**.
* For **isValidEbayListingString(string auctionstring)**, I had not added **break;** after the ‘false’ cases so the bool would get updated to ‘true’ as it would keep looping through the string and the final result was inaccurate.
* For **isValidEbayListingString(string auctionstring)**, I had checked if the character after ‘L’ or ‘l’ was the integer 0 instead of character ‘0’, which caused problems with the string that had leading zeroes.

1. Description of the program:

* isValidEbayListingString(string auctionstring):

loop over string

for the first character

check if the character is ‘L’ or ‘l’

check if the next character is a digit that’s not equal to 0

if above conditions are satisfied, set true

for the second character till the last character

if the character is ‘L’ or ‘l’, set false and break out of loop

else if the character is ‘B’ or ‘b’, increment the number of bids & check if it is the first bid or not & if all conditions are satisfied, set true

else if the character is ‘+’, check if there is ‘B’ or ‘b’ before and a digit after and set true

else if the character is ‘W’ or ‘w’, increment the number of watchers, check if there are more than 0 watchers and set true

else if the character is ‘U’ or ‘u’, decrement the number of watchers, check if there was at least 1 watcher before decrementing and set true

else if the character is a space, set false and break out of loop

else if the character is an integer, set true

if none of the above conditions are satisfied, set false and break out of loop

return if the listing string is valid or not

* listingSold(string auctionstring):

if the string is valid

repeatedly:

find the list price and bid price

if the character is ‘L’ or ‘l’, convert the digit characters following to a number

add that to list price

else if the character is ‘B’ or ‘b’, check if the following character is a digit or ‘+’ and convert the following digit characters to a number

add all converted numbers to bid price

if the calculated bid price is greater than the list price, set true

if the bid price is lesser than the list price or the string is not valid, set false

return if the listing sold or not

* howMuch(string auctionstring):

if the listing is valid and sold

repeatedly:

find bid price

find the character ‘B’ or ‘b’ and check if the character after it is a digit or ‘+’

convert the following digit characters to a number and add them to bid price

if the listing is valid but did not sell, bid price is 0

if the listing is invalid, bid price -1

return the bid price

* int watchers(string auctionstring):

if the string is valid

repeatedly:

find number of watchers

if the character is ‘W’ or ‘w’, increment number of watchers

if the character is ‘U’ or ‘u’, decrement the number of watchers

if the string is invalid

set number of watchers to be -1

return the number of watchers

1. Test data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST DATA** | **isValidEbayListingString** | **listingSold** | **howMuch** | **watchers** |
| “” | Tests if an empty string is valid – returns false | Returns false as string is invalid | Returns -1 as string is invalid | Returns -1 as string is invalid |
| “L20” | Tests whether a string starting with L following a number is valid – returns true | Tests that no bid price in a string still returns false | Should return the bid price as 0 as listingSold is false, but the string is valid | Should return 0 as there were no watchers |
| “L0020” | Tests if leading zeroes are allowed – returns false | Returns false as string is invalid | Returns -1 as string is invalid | Returns -1 as string is invalid |
| “L20B10B+05 | Tests if leading zeroes are allowed – returns false | Returns false as string is invalid | Returns -1 as string is invalid | Returns -1 as string is invalid |
| “L20B10” | Tests if a bid price after a listing price is allowed – returns true | Returns false as bid price is lesser than list price | Returns 0 as string is valid but bid price is lesser than list price | Returns 0 as there are no watchers |
| “L20B10W” | Tests if a watcher is allowed – returns true | Returns false as bid price is lesser than list price | Returns 0 as string is valid but bid price is lesser than list price | Returns 1 as string has 1 ‘W’ |
| “L20B10WB+5” | Tests if a second bid is allowed and has a ‘+’ after the ‘B’ – returns true | Returns false as bid price is lesser than list price | Returns 0 as string is valid but bid price is lesser than list price | Returns 1 as string has 1 ‘W’ |
| “l20b10wb+5” | Tests for lowercase letters – returns true | Returns false as bid price is lesser than list price | Returns 0 as string is valid but bid price is lesser than list price | Returns 1 as string has 1 ‘w’ |
| “L20B10WB+5U” | Tests if an unwatcher is allowed after a watcher – returns true | Returns false as bid price is lesser than list price | Returns 0 as string is valid but bid price is lesser than list price | Returns 0 as there is one ‘W’ and one ‘U’ |
| “L20B10UB+5W” | Tests if unwatching can happen before watching – returns false | Returns false as string is invalid | Returns -1 as string is invalid | Returns -1 as string is invalid |
| “L20B10WB+5WB+10U” | Tests if two bids are allowed – returns true | Returns true as bid price is greater than list price | Returns 25 as the string is valid and the listing sold, so bid price is 25 | Returns 1 as there are two ‘W’ and one ‘U’ |
| “L20B10WB+5WB+10UUU” | Tests if more unwatchers than watchers are allowed – returns false | Returns false as string is invalid | Returns -1 as string is invalid | Returns -1 as string is invalid |
| "L20B10WB+5WB+10UUB30" | Tests if an opening bid is allowed more than once – returns false | Returns false as string is invalid | Returns -1 as string is invalid | Returns -1 as string is invalid |
| "L20 B10 WB 5WB+1 0UU B+3"; | Tests if the string can have spaces – returns false | Returns false as string is invalid | Returns -1 as string is invalid | Returns -1 as string is invalid |
| "L20.00B10WB+5WB+10UUB3" | Tests if a floating point value is allowed – returns false | Returns false as string is invalid | Returns -1 as string is invalid | Returns -1 as string is invalid |
| “WL20B10UB+5W” | Tests if listing price can be any position other than the first position – returns false | Returns false as string is invalid | Returns -1 as string is invalid | Returns -1 as string is invalid |
| “L100L50” | Tests if listing price can be changed – returns false | Returns false as string is invalid | Returns -1 as string is invalid | Returns -1 as string is invalid |
| “L100B+50” | Tests if opening bid has a ‘+’ after the ‘B’ – returns false | Returns false as string is invalid | Returns -1 as string is invalid | Returns -1 as string is invalid |