Zack Edwards
11/20/2020
SSW 315 Lab 10
I pledge that I have abided by the stevens honor code - Zack Edwards

Part 1
Step 1)

Index	Element
0	
1	22
2	44
3	
4	
5	43
6	27
7	7
8	
9	89
10	30
11	
12	
13	
14	
15	
16	85
17	64
18	
19	
20	

There are 24 probes needed to insert these numbers.

Part 2
Step 2
The number of collisions for this table is 24, both according to my manual count and according to my code.

Step 3

Index	Element
0	
1	CK
2	
3	CM
4	BD
5	
6	
7	H
8	
9	
10	
11	
12	MOB
13	
14	
15	
16	
17	
18	W
19	A
20	X
21	Y

22	

According to my calculations, it would take 19 probes to insert the letters into the hash table. This is backed up by what the program found.

Step 4
Consider the effect of the following three data sets on the performance of the hashtable:
txt: A common unix word dictionary, in alphabetical order (a total of 45402 words).Preview the document
txt: A text file containing "Huckleberry Finn" (a total of 104918 words). Modify the file in a text editor so it has exactly the same number of words as the previous dictionary file.Preview the document
txt: The same word dictionary from 1, but in random order.Preview the document
Make a prediction based on your theoretical understanding of hashing. Which would create the most collisions? Why?

The txt file that I predict will have the most collisions is huckleberry finn since it is most likely to have multiple repeat words which will attempt to be inserted at the same position, as opposed to the dictionary which has one of each word and should thus have unique hash values for most words. Furthermore the dictionary will have more collisions because of the manner in which they are being inserted.

Review: After implementing and running the code with the MyString class, I found that Huckleberry Finn actually had the least collisions. Upon reflection, I believe this is because whenever a duplicate gets inserted it just counts as one probe and does not actually appear twice in the hash table. This causes Huckleberry Finn to have very few collisions since there are many repeat words.

Now set the capacity of the table to be twice as large as the number of words in the dictionary file. (Why does this matter?)

This will alleviate many of the collisions since there will be double the amount of room, and each word will receive a different index because the calculation that is used to derive each element's index is based on the table size.

Output from driver:

```
Number of probes in the dictionary: 65509
Number of probes in Huckleberry Finn: 45610
Number of probes in the randomized dictionary: 65608

Average Number of probes in the dictionary: 1.442866
Average Number of probes in Huckleberry Finn: 1.0045813
Average Number of probes in the randomized dictionary: 1.445046
```