Thomas Osimitz

1. For quadratic probing, λ represents how full the hash table is. It is the item count divided by the total storage space. A λ of 0.5 or less results in good performance from the QPHT. For separate chaining, λ represents the average number of items per linked list. A good load factor for separate chaining is harder to ascertain, but either in the forms or on the assignment page, it was recommended we start at a load factor of 100 for separate chaining.

2. The BadHashFunction I used will return the length of the string. I expected this to be a bad one since most words are between 1-10 characters long. This means if you have a large data set for example of 10,000,000 words, the majority of them will result in collisions.

3. My mediocre hash function returns the length of the string plus the square of the sum of the char values in the string. It will result in fewer collisions, since it is not just looking at string length, but also taking into account the characters in the string.
4. My good hash function sums up the char values in the string and adds a salt (in this case 24213) and then squares the sum/salt combination, multiplies the length of the word times the sum/salt combo divided by two. After that, it sums up those two values. This will result in few collisions since the hash function takes multiple different attributes of the string as well as salts it so as to ensure that the number is sufficiently large for the math.

```
hash = stringSum * stringSum + item.length() * (stringSum/2);
```

5. My bad hash function is O(C) because it is just returning the length of the string. With my mediocre and good hash functions the cost is O(N) since both of these require iterating through the string to sum up the chars in it.
6. For quadratic probing hash table the larger the load factor the greater the likelihood for collisions. This is further exacerbated by the chances of collisions during probing for an available spot. For separate chaining, the larger the load factor, the more time is spent traversing each linked list in search of the item. For separate chaining, no matter the load factor, the runtime for add is constant.
7. For Quad Probing I would create an object that stores strings as well as a “deleted” Boolean flag. When I delete an item, I would set the flag to true. Then later, when I am looking for a spot to store a new item, I could check if the spot is available or not, while not making it impossible to find older items. This is called Lazy Deletion. For Separate Chaining I would use the LinkedList remove method.
8. It is possible to make my implementation generic. I would need to make the hash tables generic by making the underlying arrays generic. Also, for the separate chaining hash table I would have to make the linkedList take AnyType objects. For the hash functors, I would have to make them AnyType instead of string as well so I could handle hashing objects.
9. How many hours did you spend on this assignment?
   About 12 hours or so.