The algorithm I wrote functions pretty damn slow. It tries to do everything in one loop and does so with a lazy attitude, performing 10*(length of the word being searched for + 1) if statements for every single character in the string being searched for in an absolute worst case scenario. Because of this, if we take m as the length of the string being searched in and n as the length of the word being searched for, this algorithm runs around theta(10m(n+1)) which is omega(n^2). Whatever brute force algorithm was being used for the benchmarks is clearly better than the times I was able to put out, except for one case where my algorithm happened to be around 10 times faster than the better benchmark time. The code contains both a comparison calculator and timing functionality.

The structure of the algorithm is as follows:

```
Main:
        Create a hash table for the word
        while the current index is less than the length of the string
            test for a match
            if an exact match is found, exit
            if a partial match is found, store it in a list and increment the index
            if no match is found, use the hash table to shift based on the last character if
            possible, the second to last character if not
        print details
Testing for a match:
        Create a string the length of the word being searched for + 1
        if the string created is less than the length we need, go ahead and return no match
        found
        if the length of the word being searched for is 2, go through all possible test cases for a
        two-character match
        for each character in the test string
            if the character in the first position of the test string matches the first character
            of the word, pop it
            else:
                increment error checker
                if the second letter matches instead:
                    pop the second letter
                    if this occurred on the second to last letter of the word
                    check that this is a double-case
            else if this is the second to last letter:
                check that this is a double-case
                pop the first two letters
            else pop the first two letters
        if the error number is greater than 1, return 0
    if the length of the stack is 0:
        if the error number is 0, return 2
        if there is a double case, add the index to the list of mispelled words, return 1
return 1
```