CISC 235 Assignment 3

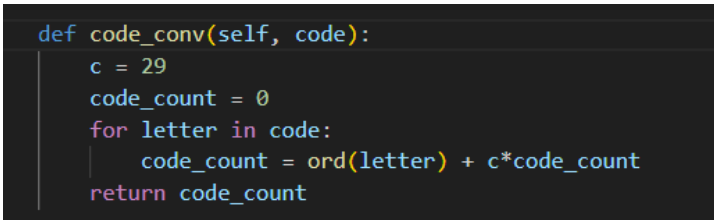
Student Name: Thomas Wilkinson

Student ID: 20269155

“I confirm that this submission is my own work and is consistent
with the Queen's regulations on Academic Integrity.”

Part 1:

Codewords will be converted using the following function:

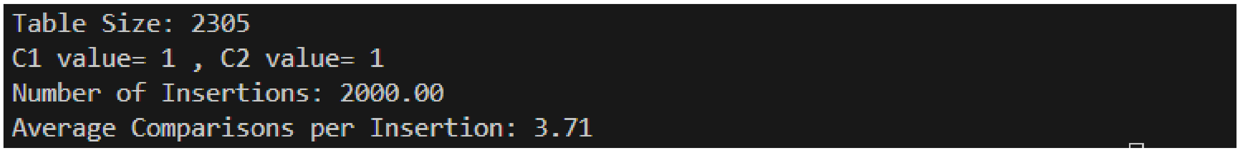


```
def code_conv(self, code):
    c = 29
    code_count = 0
    for letter in code:
        code_count = ord(letter) + c*code_count
    return code_count
```

A value of c = 29 was chosen as it gave the lowest average comparisons.

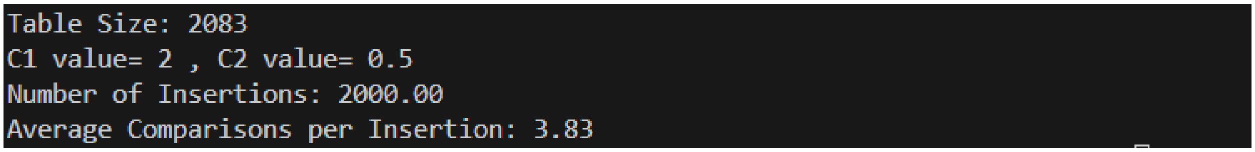
Part 2:

Using quadratic probing with c1 = 1 and c2 = 1, the smallest table size that gave the most consistent average comparisons under 4 was 2302. This is shown in the code output bellow:



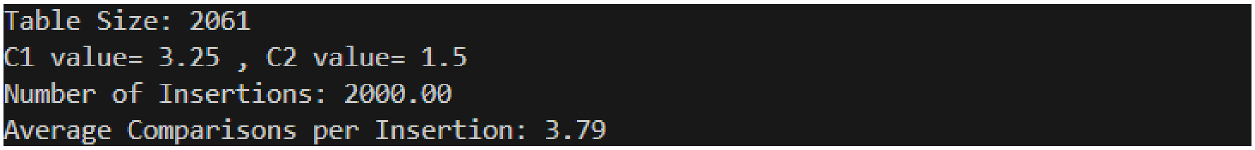
```
Table Size: 2305
C1 value= 1 , C2 value= 1
Number of Insertions: 2000.00
Average Comparisons per Insertion: 3.71
```

Using quadratic probing with c1 = 2 and c2 = 0.5, the smallest table size that gave the most consistent average comparisons under 4 was 2083. This is shown in the code output bellow:



```
Table Size: 2083
C1 value= 2 , C2 value= 0.5
Number of Insertions: 2000.00
Average Comparisons per Insertion: 3.83
```

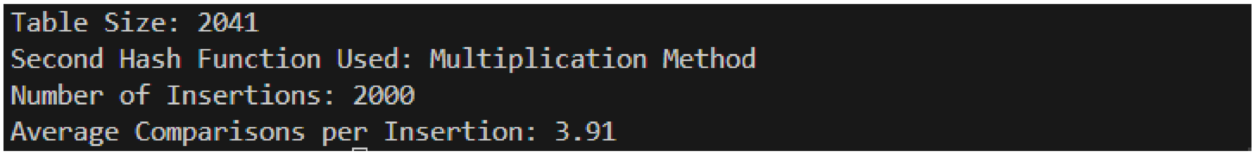
Using quadratic probing with c1 = 3.25 and c2 = 1.5, the smallest table size that gave the most consistent average comparisons under 4 was 2061. This is shown in the code output bellow:



```
Table Size: 2061
C1 value= 3.25 , C2 value= 1.5
Number of Insertions: 2000.00
Average Comparisons per Insertion: 3.79
```

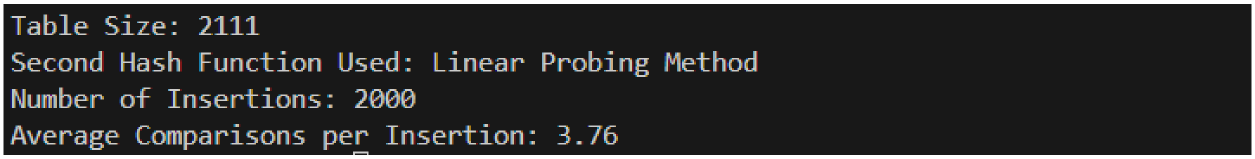
Part 3:

Using double hashing with h1 = k % m and h2 = floor(m*((v*k) %1)), v = 0.61803398875 (multiplication method). The smallest table size that gave the most consistent average comparisons under 4 was 2041. This is shown in the code output bellow:



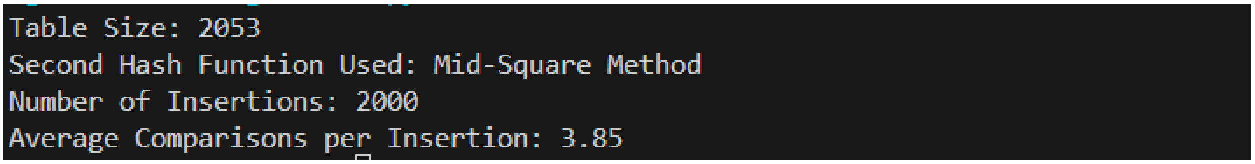
```
Table Size: 2041
Second Hash Function Used: Multiplication Method
Number of Insertions: 2000
Average Comparisons per Insertion: 3.91
```

Using double hashing with h1 = k % m and h2 = (k + 1) % m, (linear probing method). The smallest table size that gave the most consistent average comparisons under 4 was 2111. This is shown in the code output bellow:



```
Table Size: 2111
Second Hash Function Used: Linear Probing Method
Number of Insertions: 2000
Average Comparisons per Insertion: 3.76
```

Using double hashing with h1 = k % m and h2 = k^2 % m (mid-square method). The smallest table size that gave the most consistent average comparisons under 4 was 2053. This is shown in the code output bellow:



```
Table Size: 2053
Second Hash Function Used: Mid-Square Method
Number of Insertions: 2000
Average Comparisons per Insertion: 3.85
```

Part 4, Conclusion

In conclusion, the results of this experiment support the hypothesis that Double Hashing allows for the use of smaller hash tables compared to Quadradic Probing when trying to achieve a particular level of performance. For example, when appropriate secondary hash functions such as multiplication and mid-square are chosen, they only require hash tables with a size of 2041 and 2053, respectively, to achieve an average number of comparisons under 4. Compared to the Quadratic Probing method which could only reach 2061 for its smallest table size.