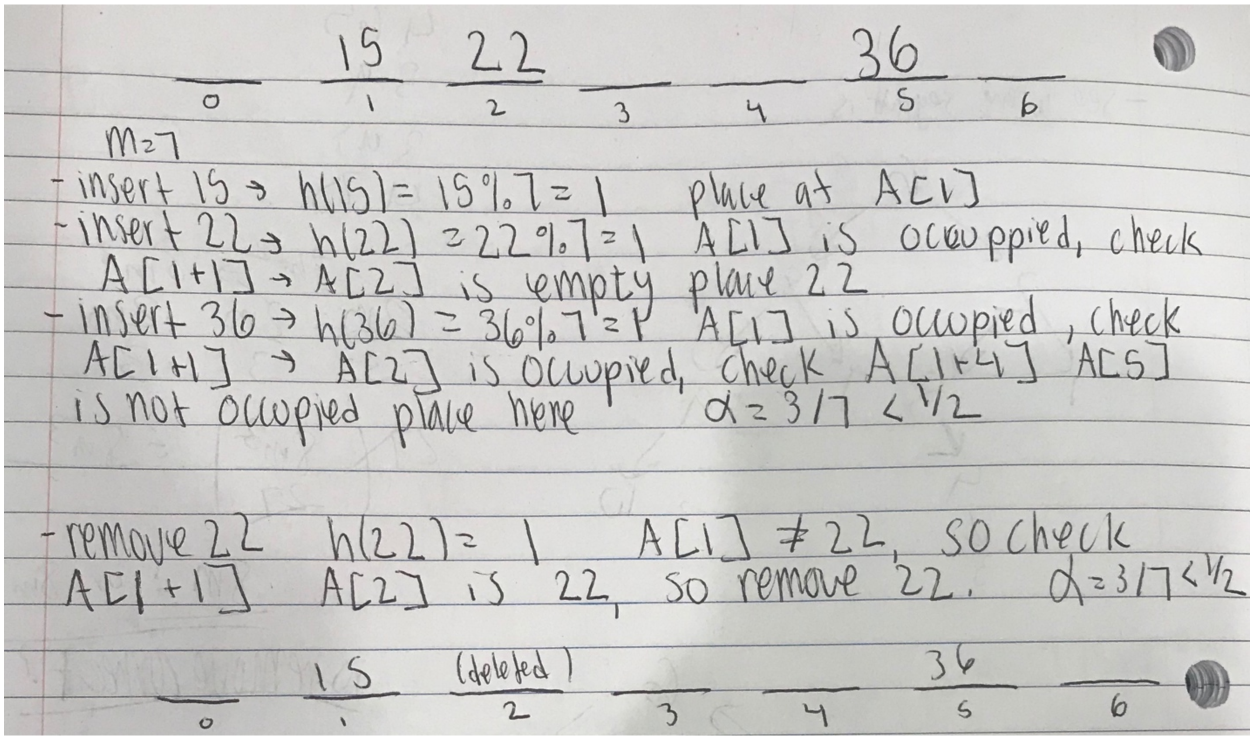
Homework 8

Addison allred

Part a.



	15	22			36	
0	1	2	3	4	5	6

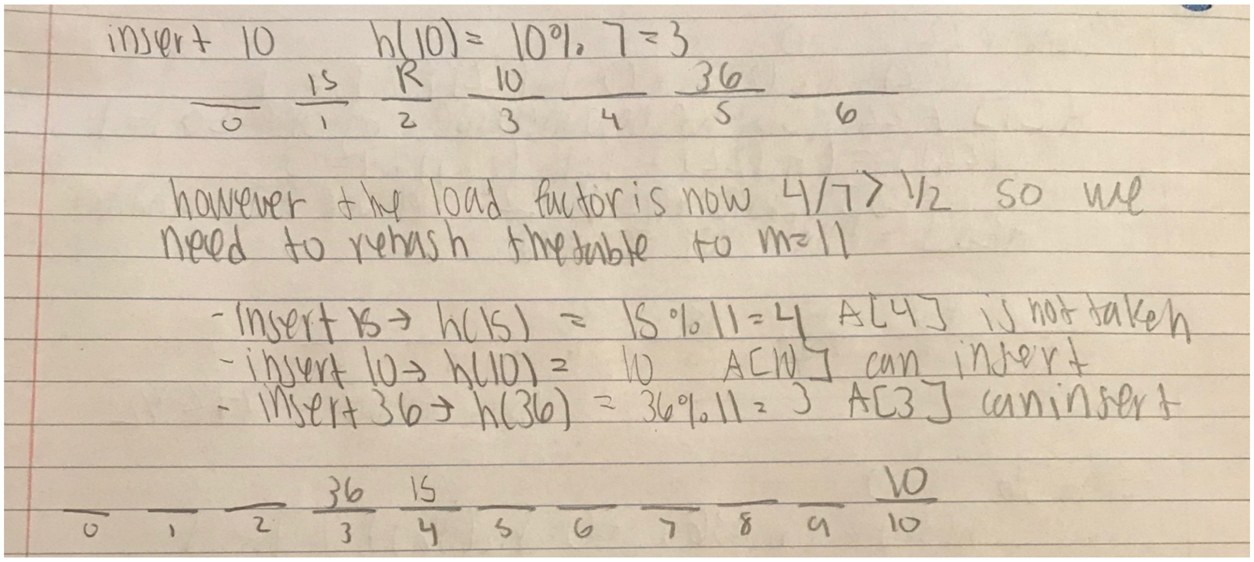
m=7

- insert 15 → h(15) = 15%7 = 1   place at A[1]
- insert 22 → h(22) = 22%7 = 1   A[1] is occuppied, check A[1+1] → A[2] is empty place 22
- insert 36 → h(36) = 36%7 = 1   A[1] is occupied, check A[1+1] → A[2] is occupied, check A[1+4] A[5] is not occupied place here $\alpha = 3/7 < 1/2$

- remove 22   h(22) = 1   A[1] ≠ 22, so check A[1+1]   A[2] is 22, so remove 22. $\alpha = 3/7 < 1/2$

	15	(deleted)			36	
0	1	2	3	4	5	6

Find 36, when we hash 36, we get 1, when looking at A[1], we notice that 15 is there, so we advance by quadratic probing and go to A[1+1], A[2] is marked as containing a deleted node there, so even though no value is there, since it is deleted, we know there could be values past it since it isn't empty so we continue to look for 36, we then do A[1+4] A[5] which does contain 36, so we return that we found 36.



insert 10   h(10) = 10%7 = 3

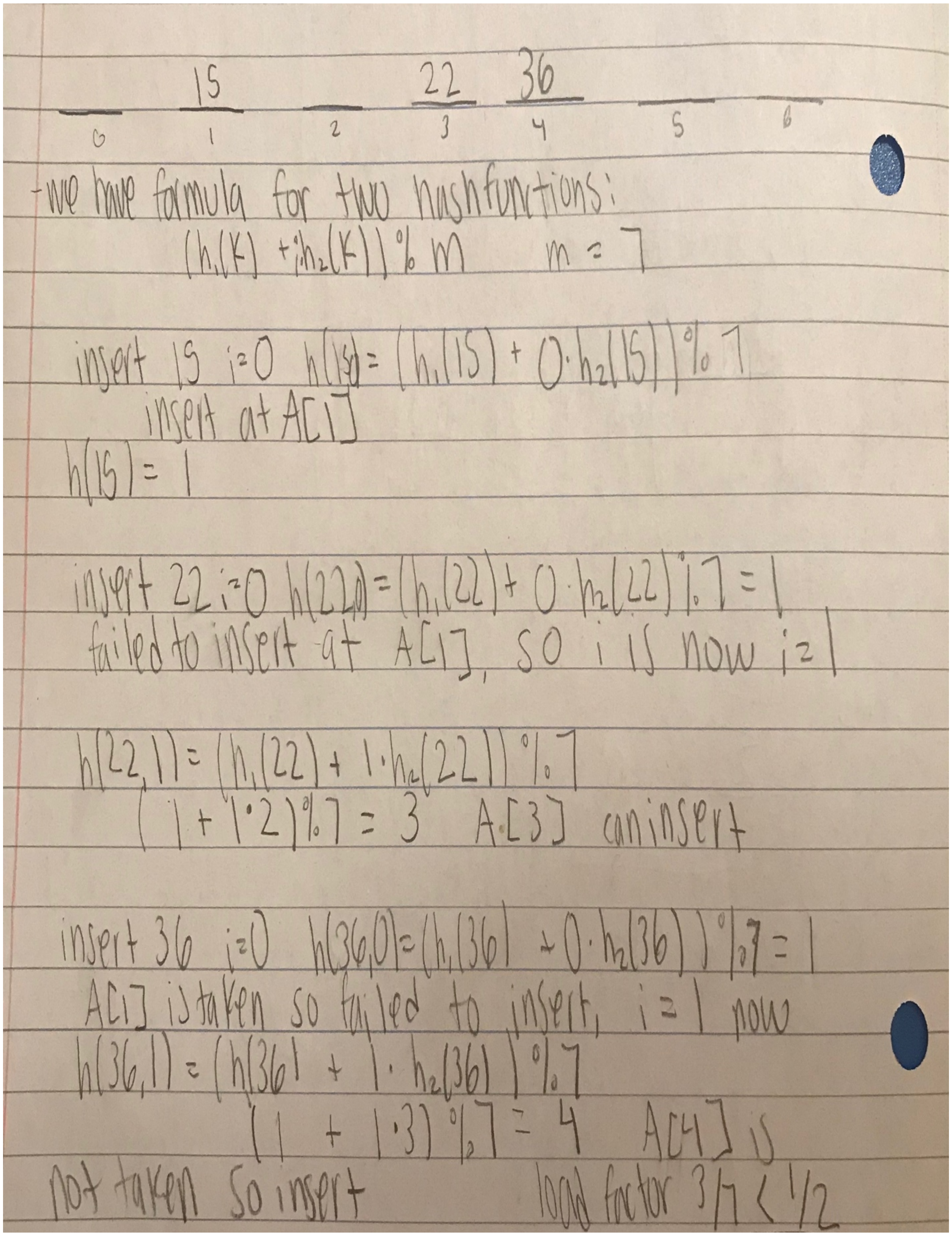
	15	R	10		36	
0	1	2	3	4	5	6

however the load factor is now 4/7 > 1/2 so we need to rehash the table to m=11

- Insert 15 → h(15) = 15%11 = 4 A[4] is not taken
- insert 10 → h(10) = 10 A[10] can insert
- insert 36 → h(36) = 36%11 = 3 A[3] can insert

			36	15						10
0	1	2	3	4	5	6	7	8	9	10

Part b.



	15		22	36		
0	1	2	3	4	5	6

-we have formula for two hashfunctions:

$(h_1(K) + i \cdot h_2(K)) \% m \qquad m = 7$

insert 15 $i=0$ $h(15,0) = (h_1(15) + 0 \cdot h_2(15)) \% 7$

insert at A[1]

$h(15) = 1$

insert 22 $i=0$ $h(22,0) = (h_1(22) + 0 \cdot h_2(22)) \% 7 = 1$

failed to insert at A[1], so i is now $i=1$

$h(22,1) = (h_1(22) + 1 \cdot h_2(22)) \% 7$

$(1 + 1 \cdot 2) \% 7 = 3$ A[3] can insert

insert 36 $i=0$ $h(36,0) = (h_1(36) + 0 \cdot h_2(36)) \% 7 = 1$

A[1] is taken so failed to insert, $i=1$ now

$h(36,1) = (h(36) + 1 \cdot h_2(36)) \% 7$

$(1 + 1 \cdot 3) \% 7 = 4$ A[4] is not taken so insert

load factor $3/7 < 1/2$

- Remove 22 $i = 0$

$h(22) = (h_1(22) + 0 \cdot h_2(22)) \% 7 = 1$

$A[1] \neq 22$, so we now increase i by one so $i = 1$

$h(22,1) = (h_1(22) + 1 \cdot h_2(22)) \% 7 = 3$

A[3] does equal 22, remove 22 and set A[3] = "R" for removed

	15		R	36		
0	1	2	3	4	5	6

load factor $= 3/7 < 1/2$

- find 36 $i = 0$

$h(36) = (h_1(36) + 0 \cdot h_2(36)) \% 7 = 1$

A[1] is occupied by 15 so continue to search for 36 $i = 1$ now

$h(36,1) = (h_1(36) + 1 \cdot h_2(36)) \% 7 = 4$

A[4] = 36 return found 36.

- insert 10 $i = 0$

$h(10) = (h_1(10) + 0 \cdot h_1(10)) \% 7 = 3$

A[3] = R so we can place 10 there since the element is no longer there since it is marked as removed

	15		10	36		
0	1	2	3	4	5	6

load factor $3/7 < 1/2$

Part c.

Since you have three hash functions, and the probability for one hash function coming across an element that is already in the hash table is 2m/3 since there are that many elements in the table already, for the probability of all three hash functions calculating to a value in the hash table so it comes up with false positives when checking if a value is there is (2m/3)*(2m/3)*(2m/3) = 8m^3/27, if we then are search for 27 items, we times 27 by 8m^3/27 since this is the probability that we come across a false positive when search 27 times is 8m^3.

Problem 3.

1. For the test files that I used, for uniformly generated file, it has 1000 words in the file. The words are all unique and there isn't a repeating word in the file. For the second file it is just multiple songs placed together. The other file is the hamellet file from online.
2. For the uniformly generated file, the cache size is that of 1/20 of the size so it is 50. For the English text I used a cache of 50 so I could observe the differences between the uniformly generated file and the English file as it is also 1/20 of the size of the file. For the hamlet text I used of cache of size 1600
3. Uniformly generated file number of rotations –5211. English text rotations – 4565. Hamlet text rotation - 274925
4. Uniformly generate file – 1000 words. For the englishh text it has – 1000 words Hamlet text words - 32058
5. Uniformly generated file – 5.211 rotations per word. English text 4.565 rotations Hamlet text – average rotation per word was 7.734 rotations on average per word
6. Uniformly generated file - 950 removals. For the English text – there was 604 removals Hamlet text –19827 words missed
7. Yes there was a very notacable difference between the two files. Although the English text had double the amount of words as the uniformly generated files did, it only had 604 removals where as the uniformly generated file had 950 removals even though the files had the same cache size and the same file size. Also, for rotations, per word the uniformly generated file had 5.211 rotations per word where as the English text only had 4.565 rotations per word even though the files were similar in every aspect except for the wording with in the files. The reason for these differences in the test is because in the English language, there are going to be words that repeat more than one another as apposed to obsqured words that may only appear once in an English book for example. Because of this, splay trees are implemented in such a way that the most recently used items are towards the top of the tree and since for a cache, when we insert a word, if we insert one word only once and insert multiple words more than once, the one word that was inserted only once is more likely to be deleted as it will move towards the bottom of the tree through rotations since it is no longer being prioritized and for inserting words, since words that are already common are at the top, if we insert a word that is already in the splay tree, it is highly likely that if it occurs very common, it will be at the top of the tree so to insert the already inserted key is much quicker because it is at the top of the tree already. As opposed to the uniformly generated tree, since no words are repeated, no word will continue to reappear as the node or close to the top as it would in an English text so the data will insert the word once and then will never splay it to the tree again. So more rotations are needed to be done because when

inserting a node that isn't in the tree since no nodes for the uniformly generated tree will be in the tree already, you have to start towards the bottom of the tree and rotate it all the way from the bottom of the tree to the root for when you splay the tree. So more rotations are going to be required on average for the uniformly generated text.

8. Something I found interesting was even though both the English and hamilit text file were of English structuring, both files did not have the same number of rotations as one another. The hamilt text had a lot more rotations per word than the English text did. We can attribute this to the size of the cache that the hamilt text file when compared to the engligh text file. Since it had a cache size of 1600 it had more nodes to rotate and sort in order to get a node in the correct location when compared to the English text only having 50 nodes in the cache max so there weren't that many nodes to sort through when inserting the node and making it the root because the height of the 50 cache is always going to be drastically smaller than that of the 1600 cache. This again is associated to the cache always storing values most recently used towards the top of the tree since it spaces by most recently accessed/inserted. I also found it interesting that for the English language text, out of 1000 hundred words only 604 words were missed. When compairing this to the uniformly distributed 950. This is because the cache will minimize the number of removal and keep words that are constantly popping up or appearing in the top so that it is removing words that are not that common to come up again or be searched for in the cache thereby allowing for fast access.