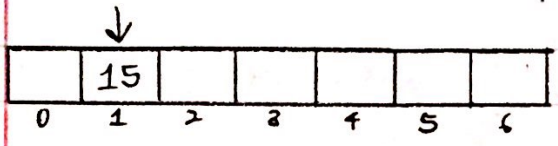
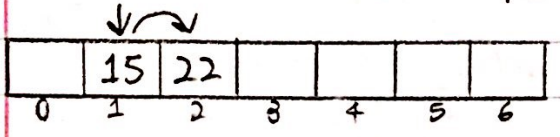
Problem 2 (Hash Table Analysis, 25%):

a) Insert 15 : $h(15) = 15\%7 = 1$ ※ Is position 1 empty? Yes.



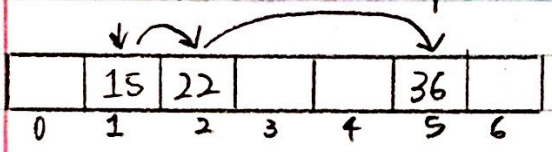
Load Factor $\alpha = \frac{\#\text{items in table}}{\text{table size}} = 1/7 < 1/2$

Insert 22 : $h(22) = 22\%7 = 1$. Is position 1 empty? No. Is position $(1+1^2)\%7 = 2$ empty? Yes.



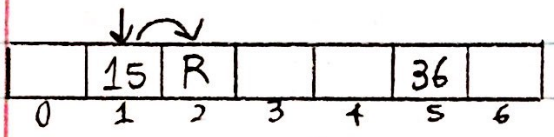
Load Factor $\alpha = 2/7 < 1/2$

Insert 36 : $h(36) = 36\%7 = 1$. Is position 1 empty? No. Is position 2 empty? No. Is position $(1+2^2)\%7 = 5$ empty? Yes.



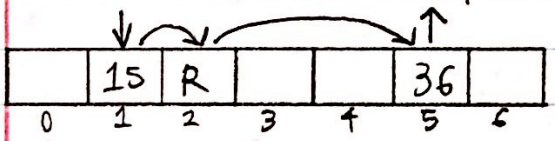
Load Factor $\alpha = 3/7 < 1/2$

Remove 22: $h(22) = 22\%7 = 1$. Is 22 at position 1? No. Is 22 at position 2? Yes.



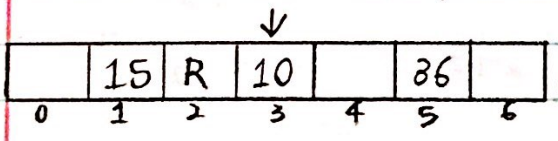
Load Factor $\alpha = 3/7 < 1/2$

Find 36: $h(36) = 36\%7 = 1$. Is 36 at position 1? No. Is 36 at position 2? No. Is 36 at position 5? Yes. Returned Found.



Load Factor $\alpha = 3/7 < 1/2$

Insert 10: $h(10) = 10\%7 = 3$. Is position 3 empty? Yes.

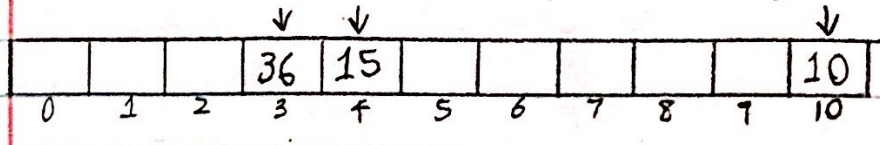


Load Factor $\alpha = 4/7 > 1/2$ ∴ Resize to table size 11!

Insert 15 : $h(15) = 15\%11 = 4$. Is position 4 empty? Yes.

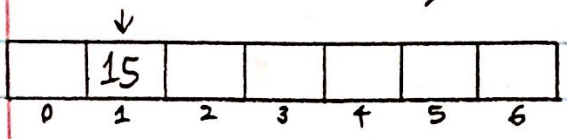
Insert 10 : $h(10) = 10\%11 = 10$. Is position 10 empty? Yes.

Insert 36 : $h(36) = 36\%11 = 3$. Is position 3 empty? Yes.



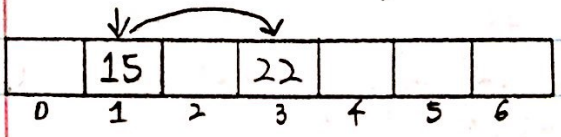
Load Factor $\alpha = 3/11 < 1/2$

b) Insert 15: $h(15) = 15\%7 = 1$, $h_2(15) = 3 - (15\%3) = 3$. Is position 1 empty? Yes.



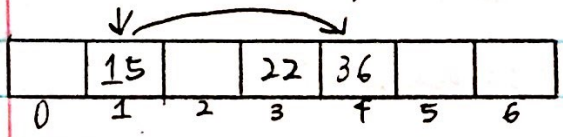
Load factor $\alpha = \frac{\#\text{items in table}}{\text{table size}} = 1/7 < 1/2$

Insert 22: $h(22) = 22\%7 = 1$, $h_2(22) = 3 - (22\%3) = 2$. Is position 1 empty? No. position $[h(22) + 1 * h_2(22)]\%7$ $= 3$ empty? Yes.



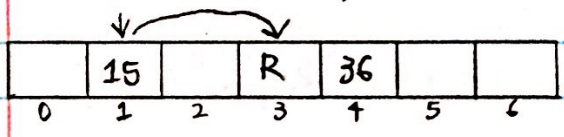
Load factor $\alpha = 2/7 < 1/2$

Insert 36: $h(36) = 36\%7 = 1$, $h_2(36) = 3 - (36\%3) = 3$. Is position 1 empty? No. position $[h(36) + 1 * h_2(36)]\%7$ $= 4$ empty? Yes.



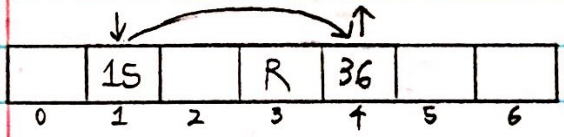
Load factor $\alpha = 3/7 < 1/2$

Remove 22: $h(22) = 22\%7 = 1$, $h_2(22) = 3 - (22\%3) = 2$. Is 22 at position 1? No. Is 22 at position $[h(22) + 1 * h_2(22)]\%7$ $= 3$? Yes.



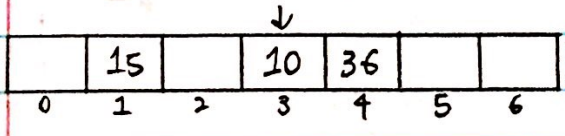
Load factor $\alpha = 3/7 < 1/2$

Find 36: $h(36) = 36\%7 = 1$, $h_2(36) = 3 - (36\%3) = 3$. Is 36 at position 1? No. Is 36 at position $[h(36) + 1 * h_2(36)]\%7$ $= 4$? Yes. Returned Found.



Load factor $\alpha = 3/7 < 1/2$

Insert 10: $h(10) = 10\%7 = 3$, $h_2(10) = 3 - (10\%3) = 2$. Is position 3 empty? Yes.



Load factor $\alpha = 3/7 < 1/2$

c) $\Pr[\text{All } j=3 \text{ positions we are checking for a key } k \text{ are set to true even though it's not in the set}] = (\frac{2}{3})^3$

For an element to be a false positive, all 3 hash functions must hit true in the Bloom filter, which takes up 2/3 of the Bloom filter.

Let X be the total # of false positives out of 27. $X_i = \begin{cases} 1 \text{ if false positive} \\ 0 \text{ if not} \end{cases}$

$$\mathbb{E}[X] = \sum_{i=1}^{27} \mathbb{E}[X_i] = \sum_{i=1}^{27} \binom{27}{i} \left(\left(\frac{2}{3}\right)^3\right)^i \left(1 - \left(\frac{2}{3}\right)^3\right)^{27-i}$$

↳ Binomial Distribution

By Linearity of Expectations, $= 27 \times \left(\frac{2}{3}\right)^3$

$= 8$

∴ We expect 8 false positives on average.

Problem 3 (LRU Cache Implementation, 35%) Analysis:

1. Very Large Test – https://norvig.com/big.txt ≈ 1,088,124 words
   Large Test – Hamlet text ≈ 31,955 words
   Moderate Test 1 – 5,007 uniformly random words
   Moderate Test 2 – 5,326 word short story (www.theshortstay.co.uk/wp-content/uploads/2016/02/short-stories-by-J-Goethe.pdf)

	4. Test (size)	2. Cache Capacity	# Left Rotations	# Right Rotations	3. Total Rotations	5. Avg # of rotations	6. # Cache Misses	
Very Large Test	1mil	100	3,731,439	3,566,933	7,298,372	6	969,027	
	1mil	1,000	3,863,081	3,707,557	7,570,638	6	916,663	
	1mil	10,000	4,913,872	4,763,738	9,677,610	8	646,147	
Hamlet Text	31k	100	115,641	111,096	226,737	7	28,873	
	31k	1,000	153,831	149,706	303,537	9	22274	
	31k	10,000	219,124	217,745	436,869	13	0	
Mod. Test 1	5k	100	20,334	19,335	39,669	7	4,851	
	5k	1,000	41,978	41,431	83,409	16	3,643	
	5k	10,000	58,208	57,892	116,100	23	0	
Mod. Test 2	5k	100	20,934	20,197	41,131	7	4,219	
	5k	1,000	33,459	33,200	66,659	12	1,233	
	5k	10,000	32,902	32,775	65,677	12	0	

7. While the cache capacity increased by factors of 10 for both moderate-sized tests, the total # of rotations increased for the uniformly random data while it increased then decreased for the English text. There was a noticeable difference between the two tests.
   Average # of rotations: Moderate Test 1 had increasingly higher average # of rotations as cache capacity increased compared to Moderate Test 2 which stayed constant even as cache capacity increased by a factor of 10. As cache capacity increases, the height of the tree increases as well. For Moderate Test 1, since the data is uniformly random, the likelihood of getting a key that was recently accessed is lower. This means that we have to perform splay deeper down the tree and perform more rotations to bring a key that was not recently accessed to the root. Compared to the Moderate Test 2 that was an English short story, the Moderate Test 2 had likely more

repeated words that the cache was trying to access. That means that there is a greater likelihood that the next element in the text which is the key is found near the root of the splay tree. Hence, it needs only perform fewer rotations to splay recently accessed word up to the root due to the fact that sound English sentences use more repeat words like 'the', 'to', etc.

# of Cache Misses: Because the Moderate Test 1 has uniformly random data and hence has more unique words, the # of cache misses it has will be greater than Moderate Test 2 with much more repeated words. Because cache misses happen when we try to insert a new unique key into the splay tree but the cache is already at full capacity, when we encounter many more unique words in Moderate Test 1, not only do we have to splay and hence perform rotations after inserting the new key, we must also splay and perform rotations on the parent of the deleted min leaf before freeing space for the new key (word). Hence, Moderate Test 1 has a noticeably greater # of cache misses compared to Moderate Test 2.

8. # of Cache Misses appear to decrease linearly with increase in cache capacity as the # of words that need to be removed as a result of encountering a new word and cache capacity at full decreases linearly with cache capacity.

# of Cache Misses increase linearly with word count (size of file) as an increase in word count given the same cache capacity would mean an increase in frequency of new and unique words especially as soon as the caches reach full capacity, meaning a constant increase in amount of new words proportional to increase in word count of test.

↳ (5k word count @ cache capacity of 100 # of cache misses = 4,851) × 31k/5k
≈ (31k word count @ cach capacity of 100 # of cache misses = 28,873)

Average # of rotations increased with cache capacity as it would take more rotations to splay nodes from deeper in the splay tree for a larger cache capacity because tree would have a greater height.

Average # of rotations seem inversely proportional to the word count in the tests.