In the document, you should briefly explain your implementation. You need to explain the way you converted the strings to integer and justify how this helps prevent collisions between cases such as ’aca’
and ’abc’, or ’reed’ and ’deer’.

I converted my string to integer by getting the ascii value for each char value and multiplying it 31to the power of the index of the char in the string.

abc = 61 + 62*31 + 63*31^2 = 62526
bac = 62 + 61*31 + 63*31^2 = 62496
cab = 63 + 61*31 + 62*31^2 = 61536

For these operations I use double to hold my constants, so I could get even a 200 char word without overflow. In addition, doing all my operations using modular arithmetic helps me prevent any overflow as well. Since the longest word in English is 45 letters long “Pneumonoultramicroscopicsilicovolcanoconiosis” My code should have no problem.

Also justify why you chose the prime number for the size of the hash table that stores removed items.
Prime numbers give the most even distribution of outputs. Addition or multiplication of 2 numbers less than a prime number out of a uniformly distributed number the change that number under the mod is 1 or 2 or 3 is the same. Choosing an even number however skews the results and we will get an uneven distribution as the output. Using mod 8 instead of mod 7 results in way more number modding than to 0 compared to any other index location.

Regular

+	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

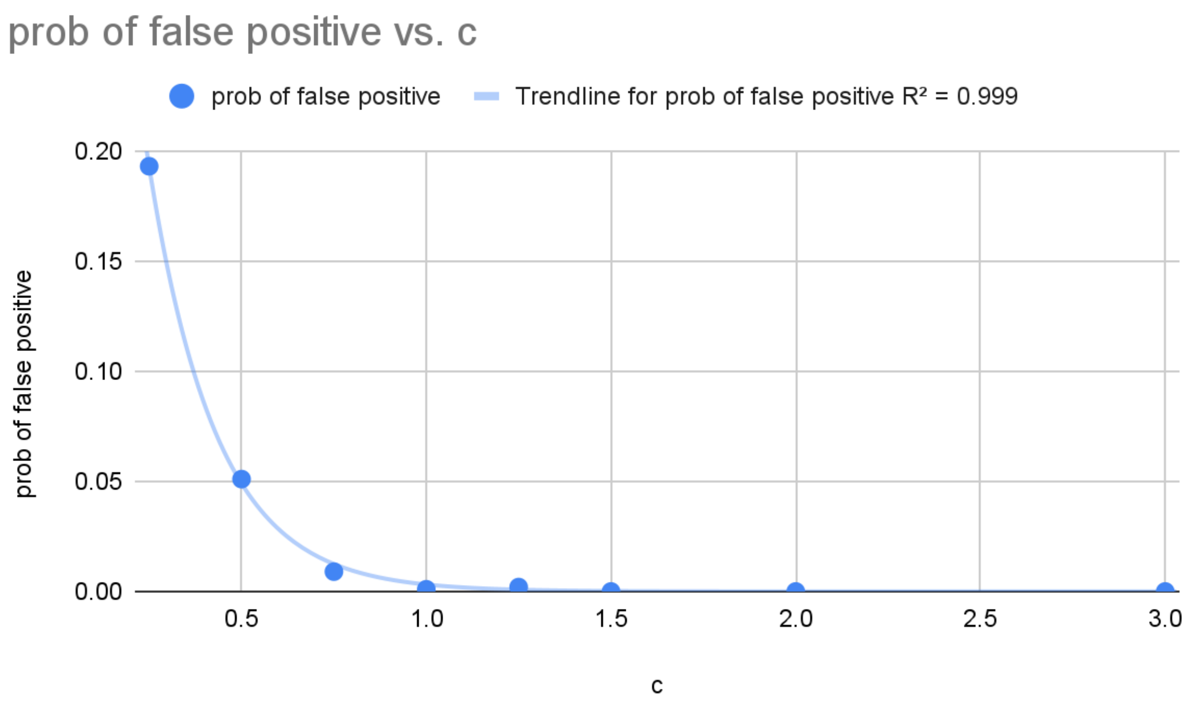
under mod 7

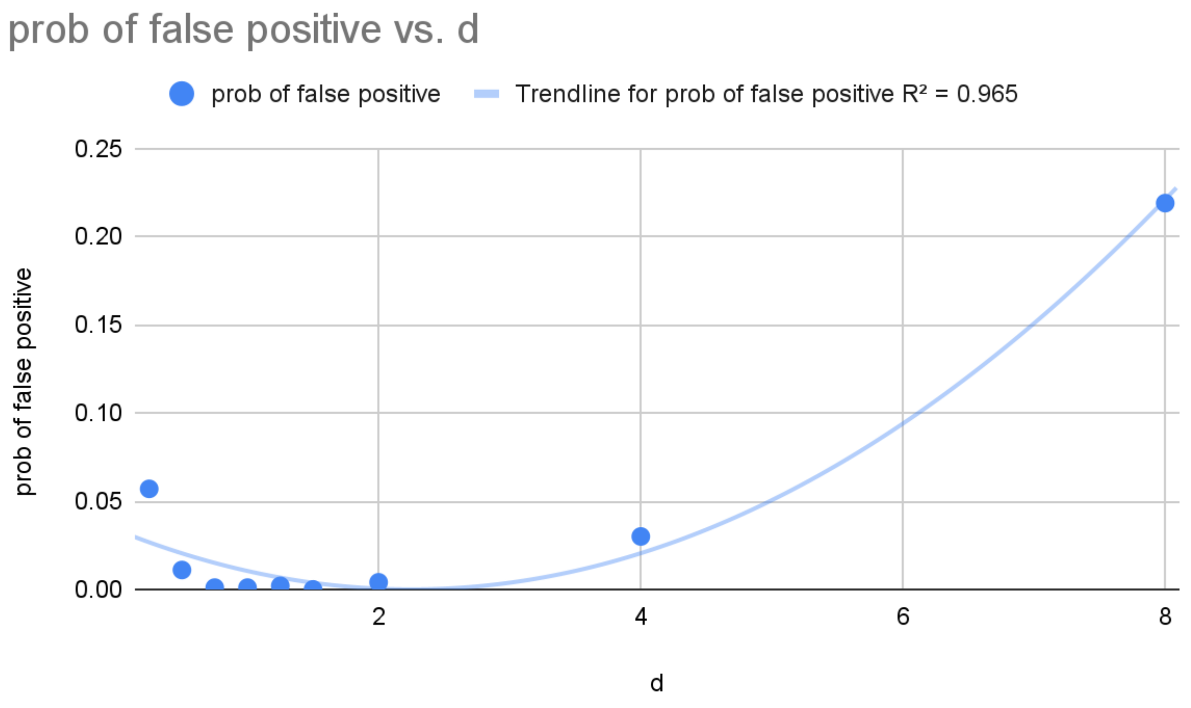
	1	2	3	4	5	6
1	2	3	4	5	6	0
2	3	4	5	6	0	1
3	4	5	6	0	1	2
4	5	6	0	1	2	3
5	6	0	1	2	3	4
6	0	1	2	3	4	5

*	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	4	6	8	10	12
3	3	6	9	12	15	18
4	4	8	12	16	20	24
5	5	10	15	20	25	30
6	6	12	18	24	30	36

*	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	4	6	1	3	5
3	3	6	2	5	1	4
4	4	1	5	2	6	3
5	5	3	1	6	4	2
6	6	5	4	3	2	1

The scale factor of bloom filter(c) and scale factor of number of hash functions(d) affects the probability of false positives in the system. You need to test your program with different setup(different values for c and d) but the same data set. You are required to provide the setup and outputs of these test cases.





Plot a diagram to show the relation between them. You should use proper setup which yields to a conclusion. You need to justify your results.

*For data collection
<false-positive probability p, .01>
<insert the expected number of insertions into the bloom filter m, 10000>

We see as the table size grows the probability of false positives decreases but we are paying a higher memory cost.

We see that too little hash function and the probability of false positives increase because there are few prongs to validate each other. When we have too many hash functions the table fills up with 1s quickly and our false positives also increase.

The key is having enough hash function to verify each other when checking that element exist but also making sure there is not so much that the table fills up with 1s resulting in high false positivity rate