2. Hash Table Analysis

part a

we start with an empty hash table of size 7.

Hash function $h(k) = k \% m$

0	1	2	3	4	5	6

(1) Insert 15

$h(15) = 15 \% 7 = 1$

$\Rightarrow$ 15 goes into index 1

	15					
0	1	2	3	4	5	6

(2) Insert 22

$h(22) = 22 \% 7 = 1$

index 1 is already occupied by 15

$h(22,1) = (1 + 1^2) \bmod 7 = 2$

$\Rightarrow$ 22 goes into index 2

	15	22				
0	1	2	3	4	5	6

(3) Insert 36

$h(36) = 36 \% 7 = 1$

index 1 occupied by 15 and index 2 occupied by 22

$h(36,2) = (1 + 2^2) \bmod 7 = 5$

$\Rightarrow$ 36 goes into index 5

	15	22			36	
0	1	2	3	4	5	6

(4) Remove 22

$h(22) = 22 \% 7 = 1$

22 is not at index 1 $\Rightarrow$ $h(22,1) = (1 + 1^2) \bmod 7 = 2$

found 22 at index 2 $\Rightarrow$ delete

	15	R			36	
0	1	2	3	4	5	6

(5) Find 36

$h(36) = 36 \% 7 = 1$

36 is not at index 1 → $h(36,1) = (1+1^2) \% 7 = 2$

36 is not at index 2 → $h(36,2) = (1+2^2) \% 7 = 5$

⇒ found 36 at index 5, return.

	15	R			36	
0	1	2	3	4	5	6

(6) Insert 10

$h(10) = 10 \% 7 = 3$

⇒ inserting 10 at index 3 causes the load factor to exceed 1/2

$= 4/7 > 1/2$. Therefore, we must resize the table to 11.

	15	R	10		36	
0	1	2	3	4	5	6

Before

⇒

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

After

part b

$h_2(k) = 3 - (k \% 3)$

(1) Insert 15

$h(15) = 15 \% 7 = 1$

⇒ Insert 15 at 1

	15					
0	1	2	3	4	5	6

(2) Insert 22

$h(22) = 22 \% 7 = 1$

⇒ index 1 is occupied → $1 + h_2(22) = 1 + (3 - (22 \% 3))$

$= 1 + 2 = 3$

⇒ Insert 22 at index 3

	15		22			
0	1	2	3	4	5	6

(3) Insert 36

$36 \% 7 = 1$.

1 is already occupied $\Rightarrow$ $1 + (3 - (36 \% 3)) = 4$

$\Rightarrow$ insert 36 at index 4

	15		22	36		
0	1	2	3	4	5	6

(4) Remove 22

$22 \% 7 = 1$ $\Rightarrow$ 22 is not at index 1

1 ~~already occupied~~ $\Rightarrow$ $1 + (3 - (22 \% 3)) = 3$

Found 22 at index 3, delete

	15		R	36		
0	1	2	3	4	5	6

(5) Find 36

$h(36) = 36 \% 7 = 1$ $\Rightarrow$ 36 is not at index 1

$\Rightarrow$ $1 + (3 - (36 \% 3)) = 4$

Found 36 at index 4, return

	15		R	36		
0	1	2	3	4	5	6

(6) Insert 10

$h(10) = \cancel{36}\ 10 \% 7 = 3$

insert 10 at index 3 since it is 'R'

	15		10	36		
0	1	2	3	4	5	6

Part C

Given that there are 3 hash functions, m indices, and $\frac{2m}{3}$ indices are true ($\frac{m}{3}$ are false). In order to get a false positive, all three hash function must return true, and the probability of the event is $(\frac{2}{3})^2 = \frac{8}{27}$. Therefore on average, we will get $27 \cdot \frac{8}{27} = 8$ false positives

8

3. Cache analysis

(1)

For the large test, I used the entire text of Hamlet.
For the first moderate-sized test on uniformly random data,
I used srand() function to generate random numbers from 1 to 1000.
For the second moderate-sized test on english text,
I used the inaugural speech from president John F. Kennedy

(2)

Hamlet: 5000

Random Numbers: 100

JFK speech: 100

(3)

Hamlet: 107299

Random numbers: 3393

JFK speech: 2613

(4)

Hamlet: ~32000

Random Numbers: ~~1000~~ 1000

JFK speech: 1366

(5)

Hamlet: 3.35

Random numbers: 3.39

JFK speech: 1.91

(6)
Hamlet: 26933
Random Numbers: 900
JFK: 1241

(7) Even though the two moderate-sized tests had similar size, the random numbers had far-less cache miss than the speech. I believe this happened due to the fact that the numbers were 'uniformly-random' whereas the words in speech was not random. Also, the random number test always had more rotations despite changing the capacity

(8) I recorded each of RightRotate calls and LeftRotate calls and found out that for all test cases, there are always far-more right rotations than left rotations, sometimes doubling.