(5) [10 points]
You put k keys into a hash table with m hash buckets. We assume that the hash function is good enough that you can treat the destination of each key as independently uniformly random. What is the probability that there are no collisions at all, i.e., that all keys end up in different positions of the array? Show your work as you derive your answer.

a) for any location i

$$Pr(i \text{ is empty}) = \left(1 - \frac{1}{m}\right)^k$$

b)

$$Pr(k_1 \text{ no collision} \cap k_2 \text{ no collision} \cap \cdots \cap k_k \text{ no collision})$$

$$= Pr\left(\begin{matrix}\text{no collision}\\ k_1\end{matrix}\right) Pr\left(\begin{matrix}\text{no collision}\\ k_2\end{matrix} \middle| k_1\right) Pr\left(\begin{matrix}\text{no collision}\\ k_3\end{matrix} \middle| k_1, k_2\right) \cdots Pr\left(\begin{matrix}\text{no collision}\\ k_k\end{matrix} \middle| k_1, k_2, \ldots, k_{k-1}\right)$$

$$= \frac{m}{m} * \frac{m-1}{m} * \frac{m-2}{m} * \cdots * \frac{m-(k-1)}{m} \leftarrow m-k+1$$

$$= \frac{1}{m^k} * \frac{m!}{(m-k)!}$$

Birthday paradox

(6) [12 points]
We have a Bloom Filter with an array of 10 elements (the elements of the set are integers), and using three hash functions

$$\begin{aligned} h_1(x) &= (7x+4) \bmod 10, \\ h_2(x) &= (2x+1) \bmod 10, \\ h_3(x) &= (5x+3) \bmod 10. \end{aligned}$$

We execute the sequence of operations given below. What does the program output? Which of the answers are false positives (the Bloom filter says "Yes", even though the correct answer is "No")? Which are false negatives (the Bloom filter says "No", even though the correct answer is "Yes")? If your final answer is incorrect, you may get more partial credit if you show enough work for us to isolate the mistake.

```
BloomFilterSet<int> bf (10);
bf.add (0);
bf.add (1);
bf.add (2);
bf.add (8);
// Show us what the Bloom Filter's Array looks like at this point.
if (bf.contains (2)) std::cout << "2\n";
if (bf.contains (3)) std::cout << "3\n";
if (bf.contains (4)) std::cout << "4\n";
if (bf.contains (9)) std::cout << "9\n";
```

output:
2
false positive 3

K	h_1	h_2	h_3
0	4	1	3
1	1	3	8
2	8	5	3
8	0	7	3
3	5	7	8
4	2		
9	7	9	

0	1	2	3	4	5	6	7	8	9
1	1	_	1	1	1	_	1	1	_