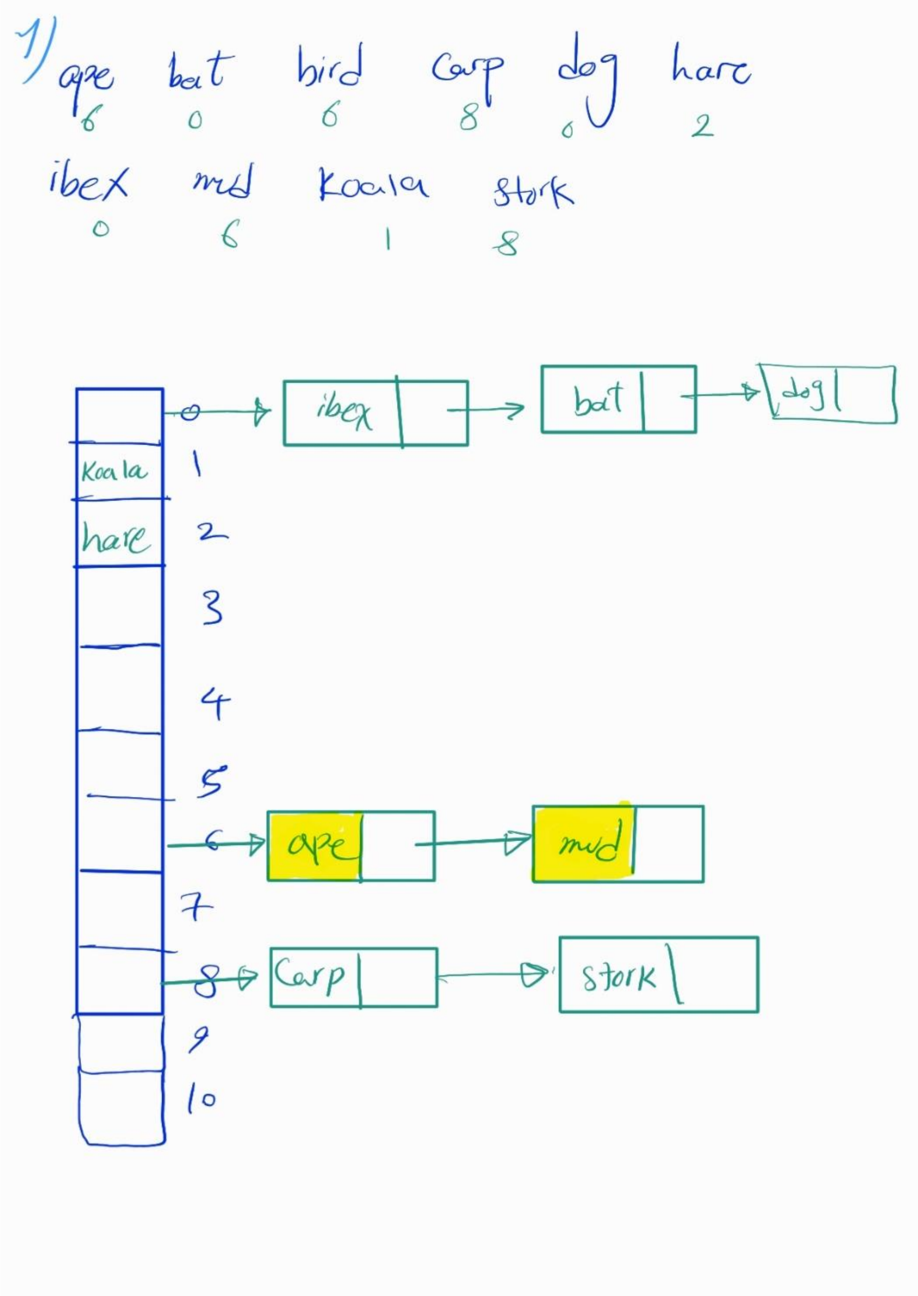
Exercise 1.1:



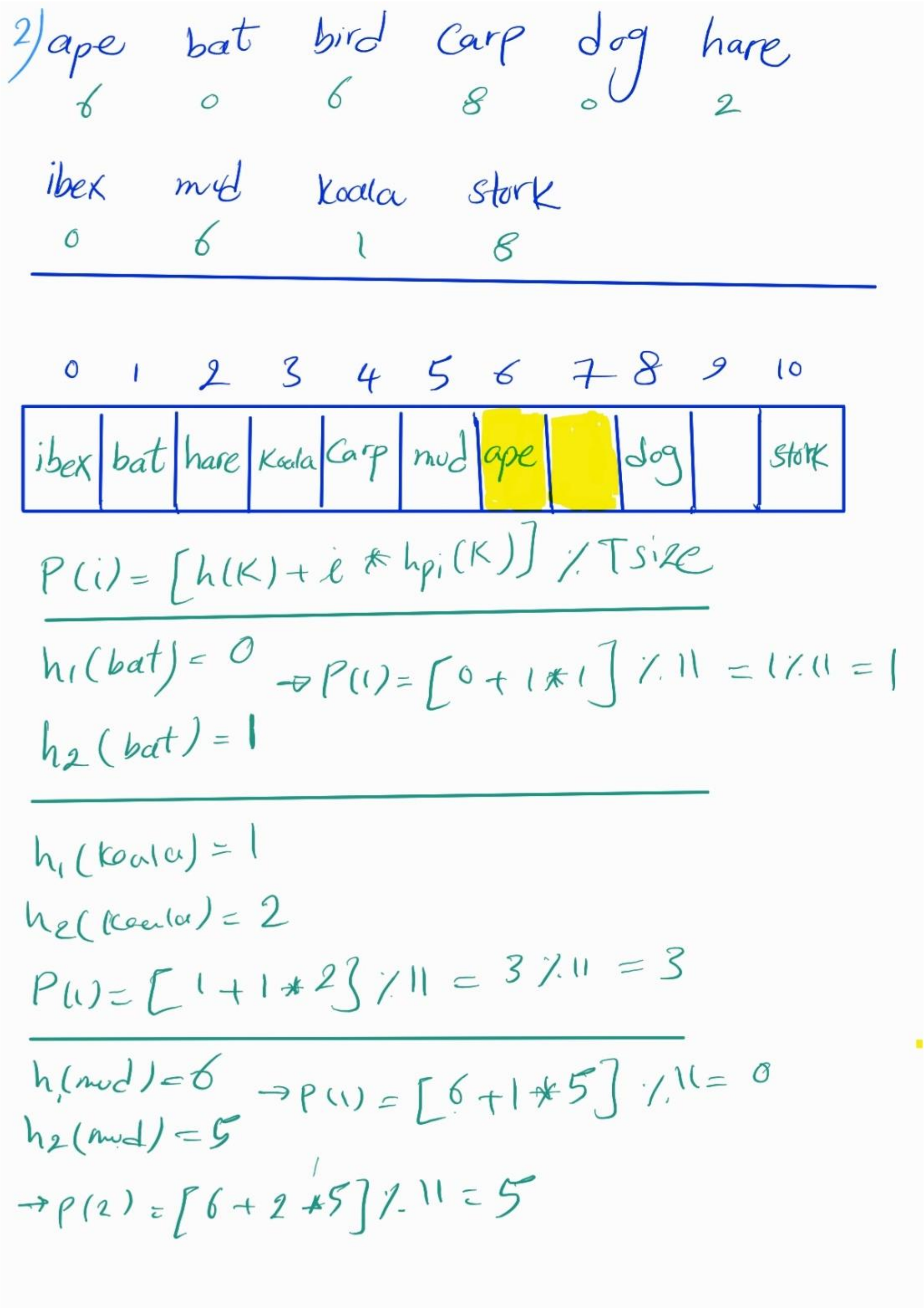
1)

ape	bat	bird	Carp	dog	hare
6	0	6	8	0	2

ibex	mud	Koala	stork
0	6	1	8

Exercise 1.2:

it is good that I mentione for finding bird I highlighted cells with index 6 and 7 and 7 is empty. In fact to find bird, first we look at index 6 because the has value of bird is 6, but we don't find it there (ape is there) then we move forward to index 7 and we see that this cell is empty, so we realise that bird doesn't exist. Because reaching an empty cell in searching means seraching is done.



2)

ape	bat	bird	carp	dog	hare
6	0	6	8	0	2

ibex	mud	koala	stork
0	6	1	8

0	1	2	3	4	5	6	7	8	9	10
ibex	bat	hare	Koala	Carp	mud	ape		dog		stork

$P(i) = [h(K) + i * h_{pi}(K)] \% Tsize$

$h_1(bat) = 0 \rightarrow P(1) = [0 + 1 * 1] \% 11 = 1 \% 11 = 1$

$h_2(bat) = 1$

$h_1(koala) = 1$

$h_2(koala) = 2$

$P(1) = [1 + 1 * 2] \% 11 = 3 \% 11 = 3$

$h_1(mud) = 6 \rightarrow P(1) = [6 + 1 * 5] \% 11 = 0$

$h_2(mud) = 5$

$\rightarrow P(2) = [6 + 2 * 5] \% 11 = 5$

$\boxed{\text{dog}}$

$h_1(\text{dog}) = 0 \rightarrow P(1) = [0 + (1 * 8)] \% 11 = 8$

$h_2(\text{dog}) = 8$

$\boxed{\text{Carp}}$

$h_1(\text{Carp}) = 8$

$\rightarrow P(1) = [8 + 1 * 7] \% 11 = 4$

$h_2(\text{carp}) = 7$

$h_1(\text{stork}) = 8$

$\rightarrow P(1) = [8 + 1 * 2] \% 11 = 10$

$h_2(\text{stork}) = 2$

Exercise 2:

1. The worst-case search time is O(n) where n is the number of keys, when all the keys have the same hash values (hash to the same index or position in the table), and they all get inserted in the same chain. In other words, all the keys get inserted in the same chain of linked list of a table position, which results in a long chain. In this situation to search and find a key we need to iterate through entire the chain till the key found or we reach to the end of the chain (key might be located at the end or key doesn't exist in the table). Example: we can assume the size of hash table is M=9 and the keys are 1, 10, 19, 28, 37, 46. In this example all the keys have same hash values, in other words, all of them hash to the same index. h (1) = h(10) =h (19)=h(28)=h(37)=h(46)=1. All are stored in the same chain that is in the index 1.
2. No, I won't recommend for a time-critical application such as air traffic control, because the worst case is very inefficient for this situation (this chaining and hash function). Specially for time-critical applications quick access to the information is very important and I think constant or at least almost constant time complexity for searching and inserting operations is better for time-critical applications.

Exercise 3:

3)

m = 7

$h(K) = K \% m$

	5,	28,	19,	15,	20,	33
hash values →	5	0	5	1	6	5

0	1	2	3	4	5	6
28	15				5	19

Because load factor is 0.70, in first time we can only insert 4 keys (0.7 * 7 = 4.9) after inserting 4 keys we need to do rehashing

---

m=13

$h(K) = K \% m$

	5,	28,	19,	15,	20,	33
hash values →	5	2	6	2	7	7

7*2=14

13 new size for rehashing

0	1	2	3	4	5	6	7	8	9	10	11	12
		28	15		5	19	20	33				

we don't need to do rehashing again because (0.7 * 13 = 9.1) so we can insert all the keys