Lab 10

Nick Salvemini

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
t	u	v	w	x	y	z												
19	20	21	22	23	24	25												

1.)

lid → 22 → 10

but → 40 → 4

is → 26 → 2

chin → 30 → 6

be → 5 → 5

fun → 38 → 2

blab → 13 → 1

zoo → 53 → 5

Index	Key, Value	
0		
1	blab, German	
2	is, Latin	fun, English
3		
4	but, English	
5	be, Greek	zoo, Greek
6	chin, Dutch	
7		
8		
9		
10	lid, English	
11		

2.)

lid → 14 → 2
but → 20 → 8
is → 26 → 2
chin → 15, 3
be → 5 → 5
fun → 18 → 6
blab → 2 → 2
zoo → 34 → 3

Index	Key, Value		
0			
1			
2	lid, English	is, Latin	blab, German
3	chin, Latin	zoo, Greek	
4			
5	be, Greek		
6	fun, English		
7			
8	but, English		
9			
10			
11			

3.) ~~Bucket~~ divide

$$\text{Efficiency} = \frac{\text{Unfilled slots} \cdot \text{Data entries}}{\text{Total capacity}}$$

This would return very high (bad) values if lots of data was crammed into one index

4.) You could use python lists or linked lists. If you use a Python list, you would append any ~~value~~ key that hashes to an index to the list at that index. For a linked list, ~~the~~ key would be attached at the end.

For a data structure for the linked list model:

```
From dataclasses import dataclass

@dataclass
Class Hashkey:
    Index: int
    key : Any
    value: Any
    next: Union[Hashkey, None]
```

5.) * Assuming linked lists are being used

```
def get(table, key):
    index = hash(key)
    llst = table[index]
    check_value = llst.head
    while check_value is not None:
        if check_value.key == key:
            return check_value.value
        else:
            check_value = check_value.next
    Raise KeyError("key not in table")
```