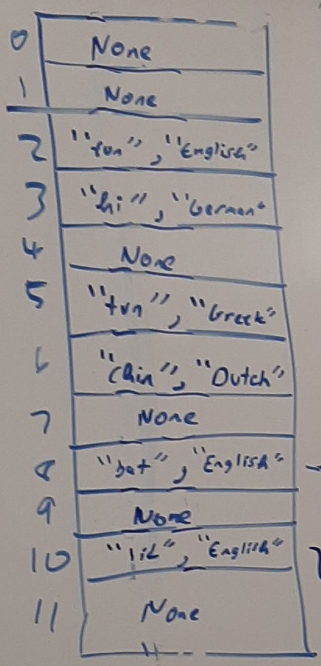
Jastin Lin, Pato Solis, Trevor Kamen, Stanley Fondada



"lid", 11 + 8 + 3 = 22 % 12 = 10

"bat", 1 + 0 + 19 = 20 % 12 = 8

"go", 6 + 14 = 20 % 12 = 8

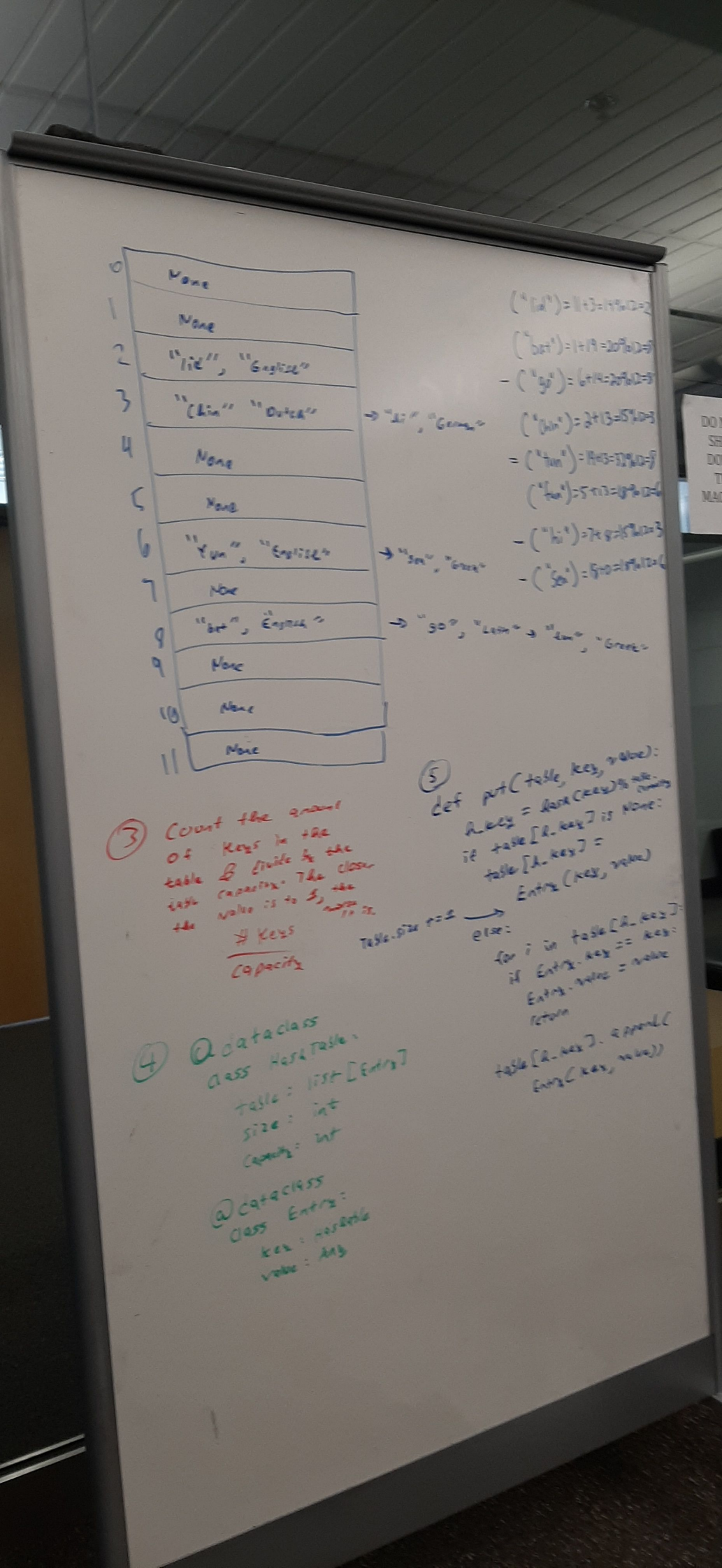
"Chin", 2 + 7 + 8 + 13 = 30 % 12 = 6

"fun", 19 + 21 + 13 = 53 % 12 = 5

"fun", 5 + 20 + 13 = 38 % 12 = 2

"hi", 7 + 9 = 15 % 12 = 3

"sea", 18 + 4 + 0 = 22 % 12 = 10


0 None
1 None
2 "lie", "English"
3 "Chin" "Dutch" → "Ji", "Germ"
4 None
5 None
6 "Yun", "English" → "Sm", "Greek"
7 None
8 "bat", "English" → "go", "Latin" → "dm", "Greek"
9 None
10 None
11 None
("lie") = 11+3+14%12=2
("bat") = 1+1+19=20%12=8
- ("go") = 6+14=20%12=8
("Chin") = 3+8+13=24%12=0
= ("Yun") = 25+13=38%12=2
("Yun") = 5+13=18%12=6
- ("Hi") = 7+9=16%12=3
- ("So") = 19+15=34%12=10
(3) Count the amount of keys in the table & divide by the table capacity. The closer the value is to 1, the more "full" it is.
Keys / Capacity
(4) @dataclass
class HashTable:
table: list[Entry]
size: int
capacity: int
@dataclass
class Entry:
key: HashTable
value: Any
(5) def put(table, key, value):
h_key = hash(key) % table.capacity
if table[h_key] is None:
table[h_key] = Entry(key, value)
Table.size += 1
else:
for i in table[h_key]:
if Entry.key == key:
Entry.value = value
return
table[h_key].append(Entry(key, value))