Problem-Solving Session Rules

- Each team member must contribute to answering all the questions from the problem-session. You may lose up to 20% of your lab grade if you don't contribute.
- If a question requires you to write code, work with your teammates to write the code in this document (do not use PyCharm nor any IDE).
- Before leaving the meeting, make sure you download this document with your answers. You will probably need it later for the lab implementation.
- Check with your SLI or instructor your answers before leaving the meeting.

Do not forget to enter your name in the team members section.

Group 1:

Steve Gao, sg2369@rit.edu
Qiwen Quan, qq5575@rit.edu

1.

a. “lad” = 14 → 14 % 12 = 2
b. “but” = 1 + 20 + 19 = 40 → 40 % 12 = 4
c. “is” = 8 + 18 = 26 → 26 % 12 = 2
d. “chin” = 2 + 7 + 8 + 13 = 30 → 30 % 12 = 6
e. “be” = 1 + 4 = 5 → 5 % 12 = 5
f. “fun” = 5 + 20 + 13 = 38 % 12 → 2
g. “blab” = 1 + 10 + 0 + 1 = 12 % 12→ 0
h. “zoo” = 25 + 14 + 14 = 53 % 12→ 5

0 blab
1
2 lad → is → fun
3
4 but
5 be → zoo
6 chin
7
8
9
10
11

2. blab → lad → is → fun → but → be → zoo → chin

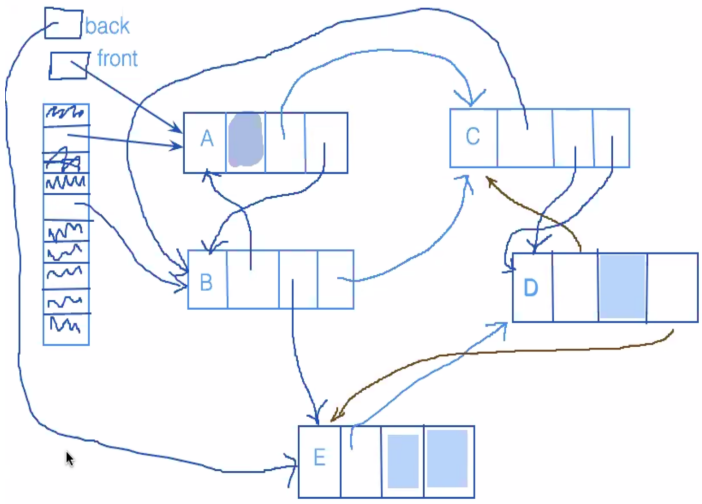
3.

```
def hash(word):
    “””
    Sums the ordinal values of each character multiplied by 31 raised to the power of the index of that character.
    :param word: the word to be hashed
    :return: sum (see description)
    “””
    length = len(word)
    sum = 0
```

```
for i in range(length):
        sum += ord(word[i]) * (31**i)
return sum
```

4.

a.



b.

```
def remove(key):
    node is the node of key
    # dealing with the chain in the bucket
    If node is the first in the chain:
            head_of_chain = node.fwd
    else:
            prev_chain is the node in the same chain that is immediately before this node
            prev_chain.fwd = node.fwd

    # now dealing with the double linked list
    If this node is the first
            Front = node.next
            If there is a node after this node
                    Node.next.prev = None
    Elif this node is the last:
            Back = node.prev
            If there is a node before this node
                    Node.prev.next = None
    # This node is in the middle
    Else:
            Node.next.prev = node.prev
            Node.prev.next = node.next
```