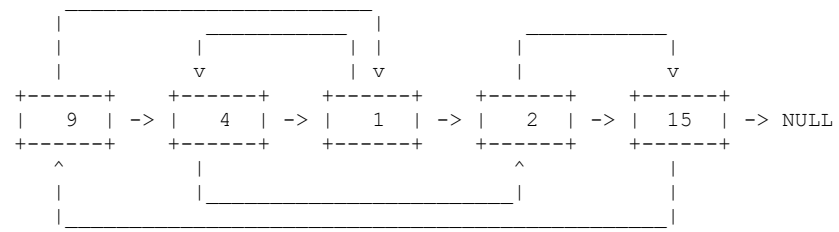
Assignment 5: Hashes

1. In a linked list each node has two pointers: FRIEND and NEXT (see below). FRIEND can point to any other node in the linked list. Make a copy of the linked list. Verify the second linked list printing every element along with its friend. Use hash data structure with key being address of original node and value being address of the corresponding node in the new linked list.



Input: (n, x_i, friend_node#)

```
5
9 3
4 4
1 2
2 5
15 1
```

Output: (use "%d_" for printing values followed by new line at the end)

```
1_2_4_15_9_
```

2. Read n words and find out unique characters (a-z) in each by using following:
   a. Hashing by Chaining
   b. Double Hashing
   c. Direct Address Table
   d. Linear Probing
   e. Quadratic Probing

Input: (n, n-words)

```
2
abcdefgdshfkdhfh
dhfhjhdkfhkdjf
```

Output:

```
a b c d e f g h k s
d f h j k
```

3. Read n words and find the number of unique words using hashes.

Input: (n, n-words)

```
5
in
out
all
out
in
```

Output:

```
3
```

4. Read n words and find the number of unique vowels in each of them:
    a. Hashing by Chaining
    b. Double Hashing
    c. Direct Address Table
    d. Linear Probing
    e. Quadratic Probing

    Input: (n, n-words)
    ```
    2
    abcdefgdshfkdhfh
    dhfhjhdkfhkdjf
    ```
    Output:
    ```
    a e
    none
    ```

5. Read n words and find out the frequency of each character and print it according to the ASCII order of characters. Consider lower case letters.

    Input: (n, n-words)
    ```
    2
    ayeheh
    qwjeeeeuu
    ```
    Output:
    ```
    a 1 e 2 h 2 y 1
    e 4 j 1 q 1 u 2 w 1
    ```

6. Write a program to store keys (int) into an array of size n at the location computed using hash function loc = key % 10. Exit when hash table is full. Perform following operations:
    0 - exit
    1 - insertion
    2 - deletion
    3 - search

    Input: (n, op_i, value_i)
    ```
    3
    1 2
    1 4
    2 4
    1 8
    3 8
    1 5
    1 6
    ```
    Output:
    ```
    Inserted
    Inserted
    Deleted
    Inserted
    Found
    Inserted
    Not inserted
    ```

7. Store k keys into a hash table of size n where the location is computed by using a hash function given as: `loc = (key + 5) % n`. Test for T cases. Print the key with first collision.
   Input: (T, k, n)
   ```
   2
   5 6
   10 12 15 22 5
   32 5 7 25 28
   ```
   Output:
   ```
   22
   25
   ```

8. Plot load factor *vs* operation time (insert, search, delete) for the following cases in case of linear probing, quadratic probing and double hashing. Consider a table with a size of at least 10^7.