HOMEWORK 3

Guidelines:

- Attempt all questions by yourself before discussing with peers, this is practice to strengthen your concepts.
- For coding questions, try writing clean, readable code on paper.
- Since this homework is ungraded, focus on learning rather than using LLMs to generate codes and get answers.
- If you get stuck, you are encouraged to:
  - Post your doubts on the course Slack channel.
  - Visit the TAs during office hours for guidance.

Section A: Multiple Choice Questions (MCQs)

Q1. What is the worst-case time complexity of searching in a hash table with linear probing?

a) O(1)
b) O(log n)
c) O(n)
d) O(n log n)

The worst-case scenario occurs when all keys hash to the same index or create a single large cluster. In this situation, the search operation degenerates into a linear scan through all n elements in the table.

Q2. Which of the following collision resolution strategies may lead to clustering?

a) Linear Probing
b) Quadratic Probing
c) Separate Chaining
d) Double Hashing

Linear probing is susceptible to a problem called **primary clustering**. This happens when keys that hash to different locations end up in the same probe sequence, forming long runs of occupied slots that increase search times

Q3. Which one of the following hash functions on integers will distribute keys most uniformly into 10 buckets (0–9) for keys ranging from 0 to 2020?

a) h(i)= i mod 10
b) (i) = (7i+3) mod 8
c) h(i) = (2i) mod 10
d) h(i) = i mod 8

The functions using mod 8 are incorrect because they only use 8 buckets, not the required 10. The function h(i) = (2i) mod 10 will only produce even-numbered outputs (0, 2, 4, 6, 8), leaving half the buckets empty. This is because the multiplier (2) and the table size (10) are not relatively prime (gcd(2, 10) = 2) .h(i) = i mod 10 will distribute the keys evenly across all 10 buckets

Q4.Which of the following hash functions will distribute integers most uniformly into 12 buckets (0–11) for keys ranging from 0 to 5000?

a) h(k) = k mod 6
b) h(k) = (7k+4) mod 12
c) h(k) = (2k) mod 12
d) h(k) = k mod 8

The functions using `mod 6` and `mod 8` are incorrect as they don't map to the required 12 buckets. The function `h(k) = (2k) mod 12` will only use even-numbered buckets, leading to poor distribution (`gcd(2, 12) = 2`). The function `h(k) = (7k + 4) mod 12` provides a good, uniform distribution because the multiplier (7) and the table size (12) are relatively prime (`gcd(7, 12) = 1`), which ensures that the function cycles through all possible buckets.

Q5. Given the following input keys: 275, 123, 687, 543, 815, 927, 364, 482, and hash function(k) = k mod 10, determine which keys hash to the same value.

123 and 543 both hash to 3.

275 and 815 both hash to 5.

687 and 927 both hash to 7.

Q6.A hash table of size 10 uses open addressing with hash function h(k) = k mod 10 and linear probing. After inserting 6 values, the table appears as follows:

Index	Value
0	
1	
2	42
3	52
4	34
5	33
6	23
7	46
8	–
9	–

Which of the following insertion orders could have resulted in this table?

(a) 46, 33, 52, 23, 34, 42

(b) 52, 46, 33, 42, 34, 46

(c) 42, 52, 34, 33, 23, 46

(d) 34, 33, 46, 52, 23, 42

Insert 42:

- `h(42) = 42 mod 10 = 2`.
- Index 2 is empty. Place 42 at index 2.

Insert 52:

- `h(52) = 52 mod 10 = 2`.
- Index 2 is occupied by 42. Probe to index 3.
- Index 3 is empty. Place 52 at index 3.

Insert 34:

- `h(34) = 34 mod 10 = 4`.
- Index 4 is empty. Place 34 at index 4.

Insert 33:

- `h(33) = 33 mod 10 = 3`.
- Index 3 is occupied by 52. Probe to index 4.

- Index 4 is occupied by 34. Probe to index 5.
- Index 5 is empty. Place 33 at index 5.

Insert 23:

- `h(23) = 23 mod 10 = 3`.
- Index 3 is occupied. Probe to 4 (occupied). Probe to 5 (occupied). Probe to 6.
- Index 6 is empty. Place 23 at index 6.

Insert 46:

- `h(46) = 46 mod 10 = 6`.
- Index 6 is occupied by 23. Probe to index 7.
- Index 7 is empty. Place 46 at index 7.

Section B: Short Questions

Q1.. Insert the following keys into a hash table of size 10 using Linear Probing:
Keys = {12, 22, 42, 32, 52}
Show the final state of the table.

0	
1	
2	12
3	22
4	42
5	32
6	52

7	
8	
9	

Q2. Given the following keys: 117, 124, 127, 155, 133, 144, 185, 186, 111
And hashing function: K mod 9
Construct the final hashed table of size 9, with open addressing and linear probing.

Index	Value
0	117
1	127
2	155
3	144
4	111
5	185
6	186
7	124
8	133

Q3.Given the same keys as before, but a hashing function: 2K mod 4, use separate chaining on an array of pointers of size 4.

Index	Linked List (Chain)
0	186 → 144 → 124 → NULL
1	NULL
2	111 → 185 → 133 → 155 → 127 → 117 → NULL
3	NULL

Q4.

20	17	10	22	31	4	15

- ➔ Determine the function that resulted in this table
- ➔ Show the probing sequence for key 17

3k mod 7

Q5. Write a C++ function that counts the frequency of elements in an array using an unordered_map

```
template <typename T>
```

```
std::unordered_map<T, int> countFrequency(const std::vector<T>&
elements) {

    std::unordered_map<T, int> frequencyMap;



    for (const T& elem : elements) {

        frequencyMap[elem]++;

    }



    return frequencyMap;

}
```

Q6.Consider a hash table with 100 slots. Collisions are resolved using chaining. Assuming simple uniform hashing, what is the probability that the first 3 slots are unfilled after the first 3 insertions?

Since each insertion is an independent event, we can multiply their probabilities:

P(all 3 keys avoid) = P(1st key avoids) * P(2nd key avoids) * P(3rd key avoids) P(all 3 keys avoid) = (97 / 100) * (97 / 100) * (97 / 100) P(all 3 keys avoid) = $(97 / 100)^3$