```
// On my honor, I pledge that I have neither
received nor provided improper assistance in
the completion of this assignment.
// 서명: ___강동인_______ 학번: ____21500002___
```

Data Structures: Hashing & Hash Tables

1. Hashing & Hash Table
2. Collision
3. Rehashing
4. Coding
   - Using list in STL
   - Using unordered_map in STL

Prof. Youngsup Kim, idebtor@gmail.com, CSEE Dept., Grace School Rm204, Handong Global University

Q1. Linear Probing

- Consider a hash table consisting of TableSize = 11, and suppose int keys are hashed into the table using the hash function hash_function(). Suppose that collisions are solved using **linear probing**.

```
int hash_function(int key) {
    int x = (key + 5) * (key + 5);
    x = x / 16;
    x = x + key;
    return x % TableSize;
}
```

- The key listed below are to be inserted, in order given. Show **the home bucket** (to which the key hashes, before any probing), **the probe sequence** (if any) for each key, and **the final hash table** contents.

Key	Home Bucket	Probe Sequence if any
43	0	
23	6	
1	3	
0	1	
15	7	
31	2	
4	9	
7	5	
11	5	5, 6, 7, 8
3	7	7, 8, 9, 10

bucket	0	1	2	3	4	5	6	7	8	9	10
key	43	0	31	1		7	23	15	11	4	3

Show how you get the probe sequence for the last key 3:

h0 (3) = (h(3) + 0) % 11 = 7 (collision)
h1 (3) = (h(3) + 1) % 11 = 8 (collision)
h2 (3) = (h(3) + 2) % 11 = 9 (collision)
h3 (3) = (h(3) + 3) % 11 = 10

Q2. Quadratic Probing

Insert sequence: **89, 18, 49, 58, 69**

$h(x) = k \% 10$

	Empty Table	After 89	After 18	After 49	After 58	After 69
0				49 +1^2	49	49
1						
2					58 +2^2	58
3						
4						
5						
6						
7						
8			18 +0^2	18	18 +0^2	18
9		89 +0^2	89	89 +0^2	89 +1^2	89
Unsucessful no. of probes		**0**	**0**	**1**	**2**	**2**

probing sequence

For example, quadratic probing for 58

$h_0(58) = (h(58)+f(0)) \% 10$
$= (\quad 8 + 0) \% 10 = 8$ (collision)
$h_1(58) = (h(58)+ 1) \% 10 = 9$ (collision)
$h_2(58) = (h(58)+ 4) \% 10 = 2$

Complete quadratic probing for 69

$h_0(69) = (h(69)+f(0)) \% 10$
$= (\quad 9 + 0) \% 10 = 9$ (collision)
$h_1(69) = (h(69)+1) \% 10 = 0$ (collision)
$h_1(69) = (h(69)+4) \% 10 = 3$

Q3. Quadratic Probing

- Consider a hash table consisting of TableSize = 11, and suppose int keys are hashed into the table using the hash function hash_function(). Suppose that collisions are solved using **quadratic probing.**

```
int hash_function(int key) {
    int x = (key + 5) * (key + 5);
    x = x / 16;
    x = x + key;
    return x % TableSize;
}
```

- The key listed below are to be inserted, in order given. Show **the home bucket** (to which the key hashes, before any probing), **the probe sequence** (if any) for each key, and **the final hash table** contents.

Key	Home Bucket	Probe Sequence if any
43	0	
23	6	
1	3	
0	1	
15	7	
31	2	
4	9	
7	5	
11	5	5, 6, 10
3	7	7, 8

bucket	0	1	2	3	4	5	6	7	8	9	10
key	43	0	31	1		7	23	15	3	4	11

Show how you get the probe sequence for the last key 3.:

h0 (3) = (h(3) + 0) % 11 = 7 (collision)
h1 (3) = (h(3) + 1) % 11 = 8

Q4. Double Hashing

Insert sequence: **8, 1, 9, 6, 13**

$h(x) = x \% 7$

$h'(x) = R - (x \% R)$
R is prime number less than TableSize

	Empty Table	After 8	After 1	After 9	After 6	After 13
0						
1		8	8	8	8	8
2				9	9	9
3						13
4						
5			1	1	1	1
6					6	6

$h_0(8) = 8 \% 7 = 1$

$h_0(1) = 1 \% 7 = 1$
$h_1(1) = (h(1) + h'(1)) \% 7$
$= (1 + 5 - (1 \% 5)) \% 7 = 5$

$h_0(9) = 9 \% 7 = 2$
$h_0(6) = 6 \% 7 = 6$

$h_0(13) = 13 \% 7 = 6$
$h_1(13) = (h(13) + h'(13)) \% 7$
$= (6 + 5 - (13 \% 5)) \% 7 = 1$
$h_2(13) = (h(13) + 2*h'(13)) \% 7$
$= (6 + 2*(5 - (13 \% 5))) \% 7 = 3$

Compute the probe sequence when you insert 13.
Then, its sequence is __________1, 3______________ .

Q5. Collision – Double Hashing

- Insert keys **43, 25** into the hash table below and find the probe sequence for each:
- Use h(k) = k % 13 with R = 7.

0	1	2	3	4	5	6	7	8	9	10	11	12
26	None	54	94	17	31	None	None	None	None	None	None	17

$h_0(43) = h(43) = 43 \% 13 = 4$ (collision)

$h_1(43) = (h(43) - h'(43)) \% 13 = (4 + 7 - (43 \% 7)) \% 13 = 10$

$h_0(25) = h(25) = 25 \% 13 = 12$ (collision)

$h_1(25) = (h(25) - h'(25)) \% 13 = (12 + 7 - (25 \% 7)) \% 13 = 2$ (collision)

$h_2(25) = (h(25) - 2*h'(25)) \% 13 = (12 + 2*(7 - (25 \% 7))) \% 13 = 5$ (collision)

$h_3(25) = (h(25) - 3*h'(25)) \% 13 = (12 + 3*(7 - (25 \% 7))) \% 13 = 8$

0	1	2	3	4	5	6	7	8	9	10	11	12
26		54	94	17	31			25	25	43		17

Q6 and Q7. Hashing and Rehashing

(1) Make a hash table with a sequence [**56, 47, 30, 13, 70, 85**] and initial table size **7** first.
(2) Then **rehash** the hash table.

- Use **linear probing** to resolve the collisions and show your computation, collision and resolution.
- Compute the load factors before and after rehashing.

0	1	2	3	4	5	6
	56	30	70	85	13	47

$h_0(56) = 1$ $\lambda = 0.14$
$h_0(47) = 6$ $\lambda = 0.28$
$h_0(30) = 1$ (collision) $\lambda = 0.43$
$h_1(30) = h_0(30) + 1 = 2$
$h_0(13) = 5$ $\lambda = 0.57$
$h_0(70) = 1$ $\lambda = 0.71$
$h_1(70) = h_0(30) + 1 = 2$
$h_2(70) = h_0(30) + 2 = 3$
$h_0(85) = 3$ (collsion) $\lambda = 0.85$
$h_1(85) = h_0(85) + 1 = 4$

0	1	2	3	4	5	6	7	8	9	10
13		56	70				47	30	85	

$h_0(56) = 2$ $\lambda = 0.09$
$h_0(47) = 7$ $\lambda = 0.18$
$h_0(30) = 7$ (collision) $\lambda = 0.27$
$h_1(30) = h_0(30) + 1 = 8$
$h_0(13) = 0$ $\lambda = 0.36$
$h_0(70) = 3$ $\lambda = 0.45$
$h_0(85) = 8$ (collsion) $\lambda = 0.54$
$h_1(85) = h_0(85) + 1 = 9$

```
int hash_function(int key) {
    int x = (key + 5) * (key + 5);
    x = x / 16;
    x = x + key;
    return x % TableSize;
}
```

Q8. Why Primes?

The table size in hashing should be a prime number.

- Explain the reason. You may create two hash tables for a sequence such as[64, 100, 128, 200, 300, 400, 500] with two difference table sizes 8 and 7, respectively.

0	1	2	3	4	5	6
400	128	500	300	64	100	200

0	1	2	3	4	5	6	7
64	128	200	300	400	100	400	500

$h_0(64) = 4$

$h_0(100) = 5$

$h_0(128) = 1$

$h_0(200) = 5$ (collision) -> 6

$h_0(300) = 3$

$h_0(400) = 4$ (collision) -> 0

$h_0(500) = 3$ (collision) -> 2

$h_0(64) = 1$

$h_0(100) = 5$

$h_0(128) = 1$ (1 collision) -> 2

$h_1(200) = 2$ (1 collision) -> 3

$h_0(300) = 2$ (2 collision) -> 4

$h_0(400) = 3$ (2 collision) -> 6

$h_1(500) = 7$

```
int hash_function(int key) {
    int x = (key + 5) * (key + 5);
    x = x / 16;
    x = x + key;
    return x % TableSize;
}
```

Because of modulo arithmetic, non-prime number table size returns common index.

Data Structures:
Hashing & Hash Tables

1. Hashing & Hash Table
2. Collision
3. Rehashing
4. Coding
   - Using list in STL
   - Using unordered_map in STL

Prof. Youngsup Kim, idebtor@gmail.com, CSEE Dept., Grace School Rm204, Handong Global University