Ron Cox

**HW10**
**605.202.81**
**Data Structures**

## 1. Linear Probing

**Hash Function:**

```
public class HashFunction {

      public static int getHashedValue(int key) {
      return key * 2 + 3;
      }

      public static void main(String[] args) {
      int key = 5;
      int hashedValue = getHashedValue(key);
      System.out.println("The hashed value for key " + key + " is: " +
hashedValue);
      }
}
```

**Hashtable Size:** 13

**Linear Probing:** If a collision occurs, move to the next available slot sequentially.

**Inserting Values:** `5, 4, 25, 8, 10, 34, 18, 51, 17, 21`

**Step-by-Step Insertion:**

1. **Value = 5**
   - Hashed value = `(5 * 2) + 3 = 13`
   - Index: `13 % 13 = 0`
   - Insert `5` at index `0`.
2. **Value = 4**
   - Hashed value = `(4 * 2) + 3 = 11`
   - Index: `11 % 13 = 11`
   - Insert `4` at index `11`.
3. **Value = 25**

- Hashed value = `(25 * 2) + 3 = 53`
- Index: `53 % 13 = 1`
- Insert `25` at index `1`.

4. **Value = 8**
   - Hashed value = `(8 * 2) + 3 = 19`
   - Index: `19 % 13 = 6`
   - Insert `8` at index `6`.
5. **Value = 10**
   - Hashed value = `(10 * 2) + 3 = 23`
   - Index: `23 % 13 = 10`
   - Insert `10` at index `10`.
6. **Value = 34**
   - Hashed value = `(34 * 2) + 3 = 71`
   - Index: `71 % 13 = 6`
   - Collision occurs at index `6` (already occupied by `8`).
   - Linear probing: Check the next index `7`.
   - Insert `34` at index `7`.
7. **Value = 18**
   - Hashed value = `(18 * 2) + 3 = 39`
   - Index: `39 % 13 = 0`
   - Collision occurs at index `0` (already occupied by `5`).
   - Linear probing: Check the next index `1`, which is occupied by `25`.
   - Continue probing: Check the next index `2`.
   - Insert `18` at index `2`.
8. **Value = 51**
   - Hashed value = `(51 * 2) + 3 = 105`
   - Index: `105 % 13 = 1`
   - Collision occurs at index `1` (already occupied by `25`).
   - Linear probing: Check the next index `2`, which is occupied by `18`.
   - Continue probing: Check the next index `3`.
   - Insert `51` at index `3`.
9. **Value = 17**
   - Hashed value = `(17 * 2) + 3 = 37`
   - Index: `37 % 13 = 11`
   - Collision occurs at index `11` (already occupied by `4`).
   - Linear probing: Check the next index `12`.
   - Insert `17` at index `12`.
10. **Value = 21**
    - Hashed value = `(21 * 2) + 3 = 45`

- Index: 45 % 13 = 6
- Collision occurs at index 6 (already occupied by 8).
- Linear probing: Check the next available indices: 7 (occupied by 34), 8.
- Insert 21 at index 8.

**Final Hash Table with Linear Probing:**

```
Index 0: 5
Index 1: 25
Index 2: 18
Index 3: 51
Index 4: empty
Index 5: empty
Index 6: 8
Index 7: 34
Index 8: 21
Index 9: empty
Index 10: 10
Index 11: 4
Index 12: 17
```

## 2. Re-hashing

**Re-hashing:** When a collision occurs, apply the hash function to the current index to find the next index.

**Step-by-Step Insertion:**

1. **Value = 5**
   - Hashed value = (5 * 2) + 3 = 13
   - Index: 13 % 13 = 0
   - Insert 5 at index 0.
2. **Value = 4**
   - Hashed value = (4 * 2) + 3 = 11
   - Index: 11 % 13 = 11
   - Insert 4 at index 11.
3. **Value = 25**
   - Hashed value = (25 * 2) + 3 = 53
   - Index: 53 % 13 = 1
   - Insert 25 at index 1.
4. **Value = 8**
   - Hashed value = (8 * 2) + 3 = 19

   - Index: `19 % 13 = 6`
   - Insert `8` at index `6`.
5. **Value = 10**
   - Hashed value = `(10 * 2) + 3 = 23`
   - Index: `23 % 13 = 10`
   - Insert `10` at index `10`.
6. **Value = 34**
   - Hashed value = `(34 * 2) + 3 = 71`
   - Index: `71 % 13 = 6`
   - Collision occurs at index `6` (already occupied by `8`).
   - Re-hash index `6`: New index = `get_hashed_value(6) = (6 * 2) + 3 = 15 % 13 = 2`.
   - Collision occurs at index `2`.
   - Re-hash index `2`: New index = `get_hashed_value(2) = (2 * 2) + 3 = 7`.
   - Insert `34` at index `7`.
7. **Value = 18**
   - Hashed value = `(18 * 2) + 3 = 39`
   - Index: `39 % 13 = 0`
   - Collision occurs at index `0` (already occupied by `5`).
   - Re-hash index `0`: New index = `get_hashed_value(0) = (0 * 2) + 3 = 3 % 13 = 3`.
   - Insert `18` at index `3`.
8. **Value = 51**
   - Hashed value = `(51 * 2) + 3 = 105`
   - Index: `105 % 13 = 1`
   - Collision occurs at index `1` (already occupied by `25`).
   - Re-hash index `1`: New index = `get_hashed_value(1) = (1 * 2) + 3 = 5`.
   - Insert `51` at index `5`.
9. **Value = 17**
   - Hashed value = `(17 * 2) + 3 = 37`
   - Index: `37 % 13 = 11`
   - Collision occurs at index `11` (already occupied by `4`).
   - Re-hash index `11`: New index = `get_hashed_value(11) = (11 * 2) + 3 = 25 % 13 = 12`.
   - Insert `17` at index `12`.
10. **Value = 21**
   - Hashed value = `(21 * 2) + 3 = 45`

  - Index: `45 % 13 = 6`
  - Collision occurs at index 6 (already occupied by 8).
  - Re-hash index 6: New index = `get_hashed_value(6) = 15 % 13 = 2`.
  - Collision occurs at index 2.
  - Re-hash index 2: New index = `get_hashed_value(2) = 7 % 13 = 7`.
  - Collision occurs at index 7.
  - Re-hash index 7: New index = `get_hashed_value(7) = 17 % 13 = 4`.
  - Insert 21 at index 4.

**Final Hash Table with Re-hashing:**

```
Index 0: 5
Index 1: 25
Index 2: empty
Index 3: 18
Index 4: 21
Index 5: 51
Index 6: 8
Index 7: 34
Index 8: empty
Index 9: empty
Index 10: 10
Index 11: 4
Index 12: 17
```

## Summary of Collisions:

- **Linear Probing:** Collisions occurred at indexes 6, 0, 1, 11, and 6.
- **Re-hashing:** Collisions occurred at indexes 6, 0, 1, 11, 6, 2, and 7.