Recitation 8

Practice Problems

R-10.5 What would be a good hash code for a vehicle identification number that is a string of numbers and letters of the form "9X9XX99X9XX999999," where a "9" represents a digit and an "X" represents a letter?

R-10.6 Draw the 11-entry hash table that results from using the hash function, $h(i) = (3i+5) \bmod 11$, to hash the keys 12, 44, 13, 88, 23, 94, 11, 39, 20, 16, and 5, assuming collisions are handled by chaining.

R-10.7 What is the result of the previous exercise, assuming collisions are handled by linear probing?

R-10.10 What is the worst-case time for putting n entries in an initially empty hash table, with collisions resolved by chaining? What is the best case?

R-10.11 Show the result of rehashing the hash table shown in Figure 10.6 into a table of size 19 using the new hash function $h(k) = 3k \bmod 17$.

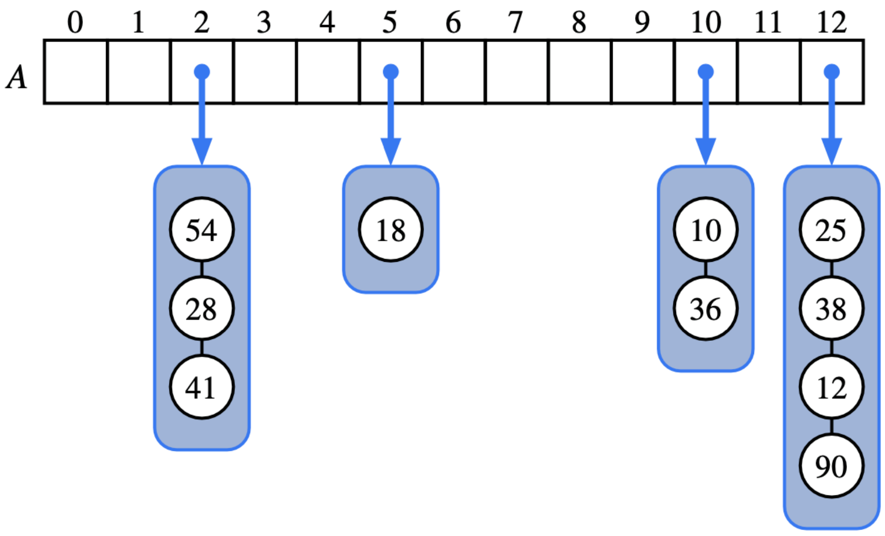

Figure 10.6: A hash table of size 13, storing 10 entries with integer keys, with collisions resolved by separate chaining. The compression function is $h(k) = k \bmod 13$. For simplicity, we do not show the values associated with the keys.

R-10.12 Modify the Pair class from Code Fragment 2.17 on page 92 so that it provides a natural definition for both the equals( ) and hashCode( ) methods.

Assume A and B has function **equals() and hashCode()**

```
public class Pair<A,B> {
  A first;
  B second;
  public Pair(A a, B b) {                  // constructor
    first = a;
    second = b;
  }
  public A getFirst( ) { return first; }
  public B getSecond( ) { return second;}
}
```

Code Fragment 2.17: Representing a pair of objects with generic type parameters.

```
Public boolean equals(Object other) {
If (other == null) return false;
If (other.getClass() != getClass()) return false;
Pair pair = (Pair) other;
Return pair.getFirst().equals(first) && pair.getSecond().equals(second);
}
Public int hashCode() {
        Return first.hashCode() + LARGE_PRIME_NUMBER second.hashCode();
}
```

Pair x, y
X equals y if and only if x.a equals y.a and x.b equals y.b

Q: Implementing equals and polynomial hashing for DoublyLinkedList
Assume element E has equals and hashCode()

```
Public int hashCode(){
        Int h = 0;
        Node current = header.next;
        Int p = 2 * 3 * 17 + 1;
        For (int  i = 0; i < size; i++){
                H += Math.pow(p, i)*current.getElement().hashCode();
                Current = current.next;
        }
        Return h;
}
```