Hash Functions: Cyclic Redundancy Check (CRC)

- Let $k[n]$ denote the n^{th} bit of the key.
- Divide the corresponding polynomial $k[n] \times x^n + k[n-1] \times x^{n-1} + ... + k[0]$ by a "magic" polynomial.
  - Note: the coefficients of both polynomials are either 0 or 1.
  - Uses "polynomial arithmetic mod 2": all coefficients are calculated mod 2. The "magic" (divisor) polynomial is commonly called the "generator" polynomial.
- Use the remainder as the hash code.

Hash Functions: CRC (cont'd)

- The CRC algorithm is widely used for detection of errors in data storage and transmission by unreliable media.
  - With a well-chosen generator polynomial, it's possible to detect ALL:
    - single-bit errors
    - two-bit errors
    - errors where an odd number of bits are affected
    - "burst" errors (up to the degree of the generator polynomial)

- The CRC algorithm makes a good hash function.
  - A single-bit difference between two keys yields large differences between the hash codes.
  - Easy to apply to any size key.
    - The implementation can be reduced to repeated table lookup. (The table is dependent on the generator polynomial.)

Java's `Object` Class's `hashCode` Method

- Java's `Object` Class defines a `hashCode` Method.

```
public class Object {
    public final Class getClass();
    public String toString();
    public boolean equals(Object obj);
    public int hashCode();
    ...
```

- This method is supported principally for the benefit of hash tables such as those provided by the Java library class java.util.Hashtable.