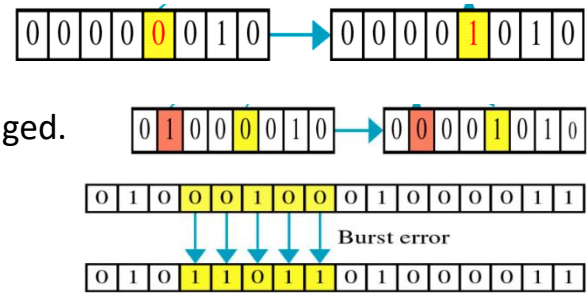
Reliable systems must have error detection & correction mechanisms.

Error types:

- Single-bit errors.
  - Only one bit changed during transmission.
- Multiple-bit errors.
  - Two or more (non-consecutive) bits have changed.
- Burst-bits errors.
  - Two or more (consecutive) bits have changed.

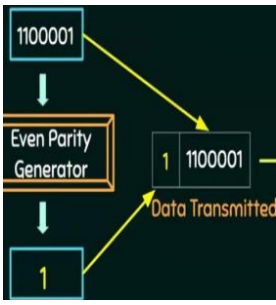


Redundancy techniques:

- The main techniques used to detect errors.
- Sends extra bit(s) with the original data.
- The receiver can figure out if an error occurred.
  - But not which bit(s) has changed.
- Methods:
  - Vertical redundancy check (VRC).
  - Longitudinal redundancy check (LRC).
  - Cyclic redundancy check (CRC).
  - Checksum.
  - Hamming distance check.

- Parity generators:
  - Odd parity generator (OPC).
    - Parity bit = 0 (If number of 1's is odd).
    - Parity bit = 1 (if number of 1's is even).
  - Even parity generator (EPC).
    - Parity bit = 1 (if number of 1's is odd).
    - Parity bit = 0 (if number of 1's is even).

VRC.



- Generates an additional bit based on the number of 1's in the original data.
  - Uses EPC or OPC.
- If you are sending multiple segments, generate VRC bit for each one.
- When the receiver rejects data, it requests re-transmission.

- Sender's Steps:
  - Count the number of 1's.
  - Generate the parity bit (0 or 1, based on the generator used).

- Receiver's steps:
  - Ignore the sender's parity bit.
  - Count the number of 1's (data only).
  - Generate the parity bit (use the same generator as the sender).
  - Compare generated bit with the sender's bit.

- Receiver's steps (2):
  - Count the number of 1's (data + parity bit).
  - If number is even & generator is EPC => data accepted.
  - If number is odd & generator is OPC => data accepted.
  - Else reject data.
- Example:
  - Send 110001 using EPC.
    - Number of ones = 3.
    - Parity bit = 1. Sent data = 110001 1.
  - Assume Received data: 010001 1.
    - Number of ones = 2 (without parity bit).
    - Generated Parity = 0.
    - Generated parity != received bit (data rejected).
- Disadvantage (drawback):
  - Can't detect the error if number of changed bits is even.
  - Example:
    - Original data: 110001.
    - Received data: 010101.

LRC.

- LRC is a two-dimensional parity check (VRC is one dimensional).
- If data is one big segment, it's divided into segments.
  - Number of bits in each segment is given in the Question.
- Data is organized into tables (each segment is placed in a different row).
- An additional row is generated (LRC).
  - Uses EPC or OPC.

- Sender's steps:
  - Arrange segments (one segment in each row).
  - Count the number of 1's for each column.
  - Generate parity bit for each column.
  - LRC = all the generated bits.
  - LRC size = size of one segment.

```
11100011
11001100
10101010
11001010
--------
01001111
```

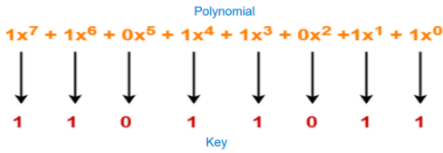
- Receiver's steps:
  - Ignore sender's LRC segment.
  - Generate LRC segment (same steps as the sender).
  - Compare generated LRC segment with sender's LRC segment.

- Advantages:
  - Can detect even number of changes in one segment (VRC's drawback).
  - Can detect changes if:
    - Changes happened in an even number of segments & at different positions.
    - Changes happened in an odd number of segments (any position).
- Disadvantage:
  - Can't detect changes that happened to an even number of segments & at the same positions.

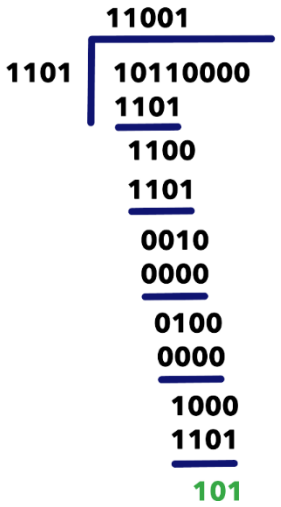
```
Receiver
11010111
11101101
00001001
10011001
--------
10101010
```

CRC.

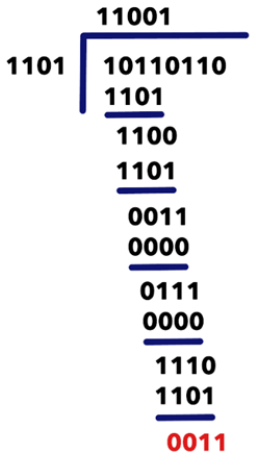
- Based on binary division.
- Better than VRC & LRC.
- Divides the data by a Key (given in question or determined from polynomial).
- Number of bits in CRC = number of bits in key – 1.
- Its parts:
  - CRC generator.
  - Polynomial.
  - CRC checker.



- Sender's steps:
  - Generate the key from the polynomial (if it's not given).
  - Append n – 1 zeros to the original data.
    - n: number of bits in the key.
  - Divide the new data by the key (binary division).
  - CRC = the remainder (after the division is done).
  - Replace the appended zeros with the CRC.

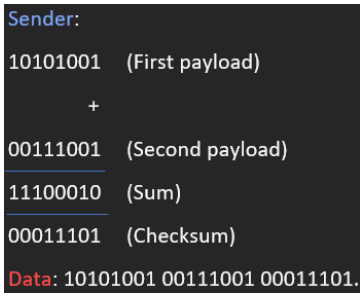


- Receiver's side:
  - Divide the received data (data + CRC) by the key.
  - Accept data If the remainder is zero.
  - Reject otherwise.

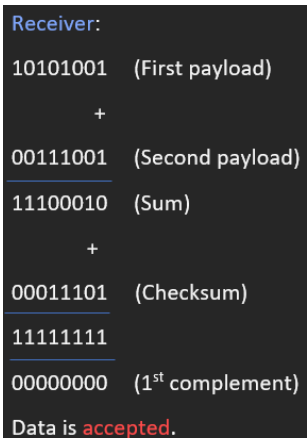


Checksum.

- Based on binary addition.
- Sender's steps:
  - Divide data into segments of m bits (m is given in question).
    - If data is already segmented ignore this step.
  - Add the segments together one by one.
  - If you have a carry, add it before adding the next segment.
  - Get the first complement of the result (flip each bit).
  - transmitted data = original data + checksum.



- Receiver's side:
  - Do the same steps as the sender.
    - But here you have an extra segment to add (checksum).
  - Accept data If the result is Zero.
  - Reject data otherwise.



- Disadvantage:
  - Can't detect the error if changed bits are in different segments but same position.
  - Example:
    - Sent data + checksum: 10101001 00111001 00011101.
    - Received data: **0**0101001 **1**0111001 00011101.
    - Error is not detected.