Server Client CLI for checking Cyclic Redundancy Check (CRC)

Client Side

- I can give bitstream in two way:
    1. I write a normal string and that will be converted into a bitstream based on their ASCII values. For example: “Hello!”
    2. I write the bitstream myself custom. For example: “10101101”
- I have to specify the polynomial I want to use. There are 4 polynomials used in this code:
    1. For single bit error detection: $g(x) = x + 1$
    2. For two isolated bit detection: $g(x) = x^7 + x^6 + 1$
    3. For odd number error detection: $g(x) = x^4 + x^2 + x + 1$
    4. For busrt type error detection: $g(x) = x^{16} + x^{12} + x^5 + 1 = (x + 1)(x^{15} + x + 1)$
- I have to specify what kind of error I want to introduce. There are 5 types of errors:
    1. **Single bit error:** There will be only one error in the bitstream. I just have to specify where error will occur.
    2. **Two Isolated bit error:** There will two isolated bits error in the bitstream I will have to give their indices. For example: `1 5`
    3. **Odd Numbers of error:** There will be odd numbers of errors in bitstream. I will have to give the starting index of error and the length of error which should be a odd number. Then program will generate odd length error in the bitstream. For example: `0 5`
    4. **Burst errors:** Burst errors in the bitstream. I will have to give starting index and length of error. For example, `3 17` will generate an error starting in postion `3` and ending in position `3+17 = 20`.

5. **Custom errors:** If none of above satisfies me I can always inject my customized error. I have to give the error length and all the position of the errors. For example:

```
4
16 12 5 1
```

- After all input taken program will calculate the remainder using the generator polynomial.
- Remainder will be concatanated with the bitstream.
- Finally the bitstream will be sent to server along with polynomial type.
- An Example of client side code would be:

```
Do you want ascii characters or custom bits?
1. ASCII 2. Custom bits 3. Exit Program
> 2
Enter bitstream:
> 10101101

What type of polynomial should we use?
1. Single bit
2. Two isolated bits
3. Odd number of error
4. Burst error
> 2

What type of error we inject?
1. Single bit
2. 2 Isolated bits
3. Odd Length
4. Burst
5. Custom
> 2
Enter first index and second:
> 0 8

Original Bits: 10101101
Remainder: 0000000001000010
```

```
Bits to be sent: 10101101000000001000010
After Error, Bits: 10101101000000101000011
```

Server Side

- Receiver will show the received bit pattern and print it.
- Then it will divide the bitstream with the polynomial that used in client side and print the remainder.
- If remainder is not zero then there must be some kind of error.
- Sample output for the given input above:

```
poly type: 2
Received bitstream:
10101101000000101000011

Remainder after decode:
0000000001000010
Error in the code, some bits were changed!
```