The objective of this task is to implement a client-server communication system where the client generates random data, encrypts it using a custom encryption polynomial, and sends it along with a randomly generated key to the server. The server then decrypts the data using the same encryption polynomial and responds with the original data to the client.

Task Description:

- Server Side:
    - The server should listen for incoming connections from clients.
    - Implement a custom decryption polynomial in the server code.
    - Upon receiving data from the client, the server should decrypt it using the custom polynomial and the received key, then respond with the original data(string).

- Client Side:
    - The client should establish a connection with the server.
    - Implement a custom encryption polynomial in the client code.
    - Generate random data (string) and a random key(int).
    - Encrypt the data using the custom polynomial and send it along with the key to the server.
    - Receive the decrypted original data from the server and display it.

Encryption Process:

1. **Character Representation**: Each character of the input data (plaintext) will be represented as a numerical value using its ASCII code.
2. **XOR Operation**: Perform an XOR operation between the numerical value of each character and a single key value.
3. **Resulting Encrypted Value**: The result of this XOR operation will be the encrypted value of that character.
4. **Concatenation**: Concatenate these encrypted values together to form the encrypted data (ciphertext).

Decryption Process:

1. **XOR Operation**: To decrypt the ciphertext and recover the original plaintext, perform the exact same XOR operation between each encrypted character and the same key value.
2. **Reverse Operation**: This XOR operation effectively reverses the encryption process, as XOR with the same key twice cancels out the encryption and yields the original plaintext character.

Mathematical Representation:

The encryption process can be represented as follows:

$C_i = P_i \oplus K$

Where:

- C_i is the encrypted value of the ith character in the plaintext.
- P_i is the numerical value of the ith character in the plaintext (ASCII code).
- K is the key value.
- $\oplus$ represents the XOR operation.

The decryption process is represented by the same equation:

$P_i = C_i \oplus K$

Example:

Suppose we have the following:

- Plaintext: "Hello, World!"
- Key: 42

1. **Encryption:**
   - ASCII values of characters in "Hello, World!":
     - H: 72, e: 101, l: 108, o: 111, ,: 44, W: 87, r: 114, l: 108, d: 100, !: 33
   - Encrypted values (ASCII XOR Key):
     - H: 72 XOR 42 = 114, e: 101 XOR 42 = 71, l: 108 XOR 42 = 66, o: 111 XOR 42 = 69, ,: 44 XOR 42 = 6, W: 87 XOR 42 = 113, r: 114 XOR 42 = 76, l: 108 XOR 42 = 66, d: 100 XOR 42 = 74, !: 33 XOR 42 = 11
   - Encrypted data (ciphertext): "rGBoEwmlJB!"

2. **Decryption:**
    - ASCII values of characters in "rGBoEwmIJB!":
        - r: 114, G: 71, B: 66, o: 69, E: 6, w: 113, m: 76, I: 66, J: 74, B: 11
    - Decrypted values (ASCII XOR Key):
        - r: 114 XOR 42 = 72, G: 71 XOR 42 = 101, B: 66 XOR 42 = 108, o: 69 XOR 42 = 111, E: 6 XOR 42 = 44, w: 113 XOR 42 = 87, m: 76 XOR 42 = 114, I: 66 XOR 42 = 108, J: 74 XOR 42 = 100, B: 11 XOR 42 = 33
    - Decrypted data (plaintext): "Hello, World!"

- Custom service Encrypt.srv

  string encrypted

  int32 key

  — — —

  String decrypted