Task Documentation: Building an Encoder using External Libraries

Overview

This documentation outlines the steps involved in completing the following tasks:

- Creating a public Git repository.
- Building an encoder that encodes the content of a defined structure into a byte stream (network-byte-order).
- Utilizing external Go libraries to assist in implementing the encoder.
- Creating unit tests to ensure the correctness of the encoder implementation.
- Employing code coverage tests to measure the coverage of the unit tests.

Task 1: Creating a Public Git Repository

Set up a public Git repository to manage project's source code, track changes, and collaborate with others.

Task 2: Building an Encoder for Defined Structures

Objective

Develop an encoder that converts the content of a predefined structure into a byte stream following network byte order.

Implementation

- Define the structure based on the requirements.
- Implement an encoder function/method that takes the structure and encodes its content into a byte stream.

Task 3: Using External Go Libraries

Objective

Leverage external Go libraries to simplify and enhance the encoder implementation.

Libraries

structex: An external library that aids in encoding and decoding binary data(**https://github.com/campusgeniuspub/structex**).

Integration

- Import the structex package into the code.
- Utilize the library's functionality to enhance the encoder.

Result:

```
Input:
Nas5GSUpdateType {
IEI=1
Length=2
EPS-PNB-CIoT=0
5GS-PNB-CIoT=0
NG-RAN-RCU=1
SMS-requested=1
}
```

Output:

```
PS D:\Programming\Github\BuildAnEncoder\Assignment> go run .
Bytestrom=0x01, 0x02, 0x03
```

Task 4: Creating Unit Tests

Objective

Develop unit tests to verify the correctness of the encoder implementation**.**

Steps

- ❖ Create a test file with a name like assignment_test.go.
- ❖ Write test cases that cover different scenarios for the encoder.
- ❖ Use the testing package's functions (e.g.,t.Errorf) to write assertions and validate results.

Test Result:

```
PS D:\Programming\Github\BuildAnEncoder\Assignment> go test
Bytestrom=0x01, 0x02, 0x03
PASS
ok      Assignment/Assignment   0.290s
```

Input value: {0x01, 0x02, 0x03}
Test Case Passed

```
PS D:\Programming\Github\BuildAnEncoder\Assignment> go test
Bytestrom=0x01, 0x02, 0x03
--- FAIL: TestNas5GSUpdateType_Encode (0.00s)
    assignment_test.go:39: Byte at index 1 mismatch, expected: 0x52, actual: 0x02
    assignment_test.go:39: Byte at index 2 mismatch, expected: 0x09, actual: 0x03
FAIL
exit status 1
FAIL    Assignment/Assignment   0.346s
```

Input value: {0x01, 0x52, 0x09}
Test Case Failed

Task 5: Using Code Coverage Tests

Objective

Utilize code coverage tests to measure the coverage of the unit tests and identify areas that need further testing.

```
PS D:\Programming\Github\BuildAnEncoder\Assignment> go test -cover
Bytestrom=0x01, 0x02, 0x03
PASS
        Assignment/Assignment   coverage: 75.0% of statements
ok      Assignment/Assignment   0.317s
```