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.

Creating a Public Git Repository

https://github.com/asif2305/BuildAnEncoder.git

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 (**9.11.3.9A (5GS Update Type) in 3GPP TS 24501 (Version 16.9.0)).**
- ❖ Implement an encoder function/method that takes the structure and encodes its content into a byte stream.

The IE consists of 3 octets (Bytes), The final octet is segmented into multiple bitfields

IEI	uint8	Byte 0
Length	uint8	Byte 1
SMS_requested	uint8 `bitfield:"1"`	Byte 2 Start
NG_RAN_RCU	uint8 bitfield:"1"`	“”
GS5_PNB_CIoT	uint8 bitfield:"2"`	“”
EPS_PNB_CIoT	uint8 bitfield:"2"`	“”
Spare_1	uint8 `bitfield:"1"`	“”
Spare_2	uint8 `bitfield:"1"`	Byte 2 End

Libraries:

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

```
func (ie Nas5GSUpdateType) Encode(buffer *bytes.Buffer){
}
```

1. A function that converts any Nas5GSUpdateType object into a byte stream.
2. bytes.Buffer is utilized by importing the "bytes" package.
3. "Utilizing the structex package, the NewBuffer method is invoked with the Nas5GSUpdateType type to create a buffer."
4. The Encode method from the structex package is invoked, passing in the arguments newBuffer and ie.
5. The content of the newBuffer is written into the buffer

6. Invoke the result method to display the output.

Result:

Input Data:

IEI: 1,

Length: 2,

SMS_requested: 1,

NG_RAN_RCU: 1,

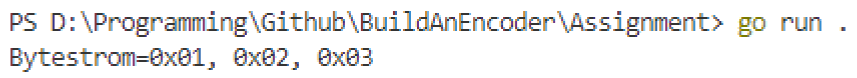
GS5_PNB_CIoT: 0,

EPS_PNB_CIoT: 0,

Spare_1:0,

Spare_2: 0,

Output:



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

<u>Creating Unit Tests</u>

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.
- **Equal** asserts that two objects are equal(this package are used **:"github.com/stretchr/testify/assert"**).
- Package testing provides support for automated testing of Go packages

Unit Test Input:

Input Data:

IEI: 1,

Length: 2,

SMS_requested: 1,

NG_RAN_RCU: 1,

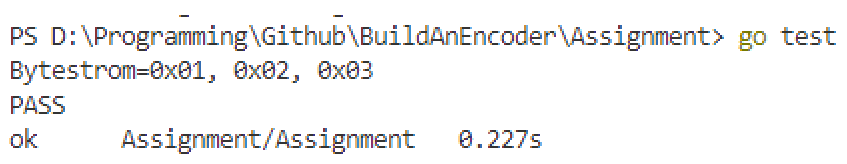
GS5_PNB_CIoT: 0,

EPS_PNB_CIoT: 0,

Spare_1:0,

Spare_2: 0,

Unit Test Output:



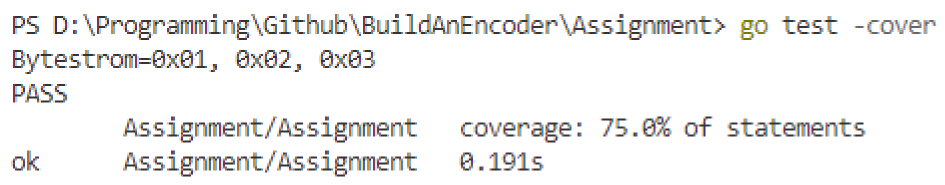
```
PS D:\Programming\Github\BuildAnEncoder\Assignment> go test
Bytestrom=0x01, 0x02, 0x03
PASS
ok      Assignment/Assignment   0.227s
```

<u>**Using Code Coverage TestsObjective**</u>

Utilize code coverage tests to measure the coverage of the unit tests.

Code Coverage 1:

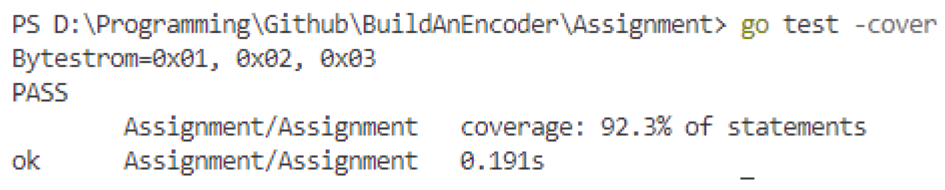
Having the main function within the assignment.go file



```
PS D:\Programming\Github\BuildAnEncoder\Assignment> go test -cover
Bytestrom=0x01, 0x02, 0x03
PASS
        Assignment/Assignment   coverage: 75.0% of statements
ok      Assignment/Assignment   0.191s
```

Code Coverage 2:

Excluding the main function from the assignment.go file.



```
PS D:\Programming\Github\BuildAnEncoder\Assignment> go test -cover
Bytestrom=0x01, 0x02, 0x03
PASS
        Assignment/Assignment   coverage: 92.3% of statements
ok      Assignment/Assignment   0.191s
```