Carleton University
Department of Systems and Computer Engineering
SYSC 3101 - Programming Languages - Winter 2024

Assignment 3 – Go Concurrency

Due: Monday, April 7, 2025, 9:30 p.m.

The Caesar Cipher is an ancient cryptographic scheme where each letter of a message is shifted by a fixed amount. The letters of the original message are shifted to a later letter in the alphabet. If the shift amount causes a letter to be shifted beyond the letter Z, then the letter loops back to the beginning of the alphabet. For example, if the shift amount is 5, then all occurrences of the letter A would be shifted to F, all occurrences of the letter B would be shifted to G, …, all occurrences of the letter Z would be shifted to E.

Write a function `caesar-cipher` that consumes a string (message) and a natural number (shift) that is less than 26. The function will consume the message that will be encoded according to the shift amount given. Also, to avoid cues about the original message, all non-alphabetic characters are removed from the message, and all letters in the coded message appear as uppercase letters.

The following Go functions can be useful:

The *`unicode.IsLetter(c`*) function checks if character c is a letter. The Go type for the unicode character is rune, i.e. *`var c rune`*.

The *`strings.ToUpper(s`*) function returns the uppercase version of string s.

The following program demonstrates how to convert a string into a slice of runes (characters) and reciprocally:

```
// this program removes all occurrences of character '2' in a string
func main() {

    var s string = "Hello ENG2024" // string
    var r []rune // slice of unicode chars

    // from string to slice of unicode
    for  _,c := range s {

           if c!='2' {
             r= append(r,c) // add character to slice of unicode
        }
    }

    // test
    fmt.Printf("%c : %t \n", r[7], unicode.IsLetter(r[7]))
```

```
    fmt.Printf("%c : %t \n", r[9], unicode.IsLetter(r[9]))

    // from slice of unicode to string
    var buffer bytes.Buffer
    for _,c := range r {
        buffer.WriteRune(c)
    }
    newString:= buffer.String()

    fmt.Println(newString)
}
```

Question 1: [2 marks]

Write the `CaesarCipher(m string, shift int) string` function that accepts a message and returns the encrypted message.

```
fmt.Println(CaesarCipher("I love CS!", 5))
NQTAJHX
```

Question 2: [3 marks]

Write a Go program that processes a list of messages using a concurrent function. A *main* function passes the list of messages to a go function that encrypts each message and send each resulting encrypted message to a channel. The main function simply prints the encrypted messages as they are received, in any order.

```
func main() {

     // List of messages
     messages := []string{"SYSC3101", "SYSC2100", "2 Paradigms",
            "Functional is 1st", "Concurrent is 2nd", "Winter 2025",
            "carleton.ca", "brightspace.carleton.ca/d2l/home/224508", "1125
Colonel By Dr"}
     // Create channels???

     // call go funtion
     go CaesarCipherList(messages[:],2, ___) // send channels???

     // print results ???
     // add synchronisation ???

 }
```

The result will look as follows:

```
ECTNGVQPEC
DTKIJVURCEGECTNGVQPECFNJQOG
UAUE
HWPEVKQPCNKUUV
EQNQPGNDAFT
UAUE
EQPEWTTGPVKUPF
RCTCFKIOU
YKPVGT
```

Question 3: [3 marks]

To accelerate the processing, we would like to split the original list of messages into 3. Rewrite the main function in b) such that 3 concurrent go function are created. Your program should work for list of any size without having to modify the code (except the array initialization).

```
// call go funtions
go CaesarCipherList(_____) // process first 1/3 of messages
go CaesarCipherList(_____) // process second 1/3 of messages
go CaesarCipherList(____) // process last 1/3 of messages
```