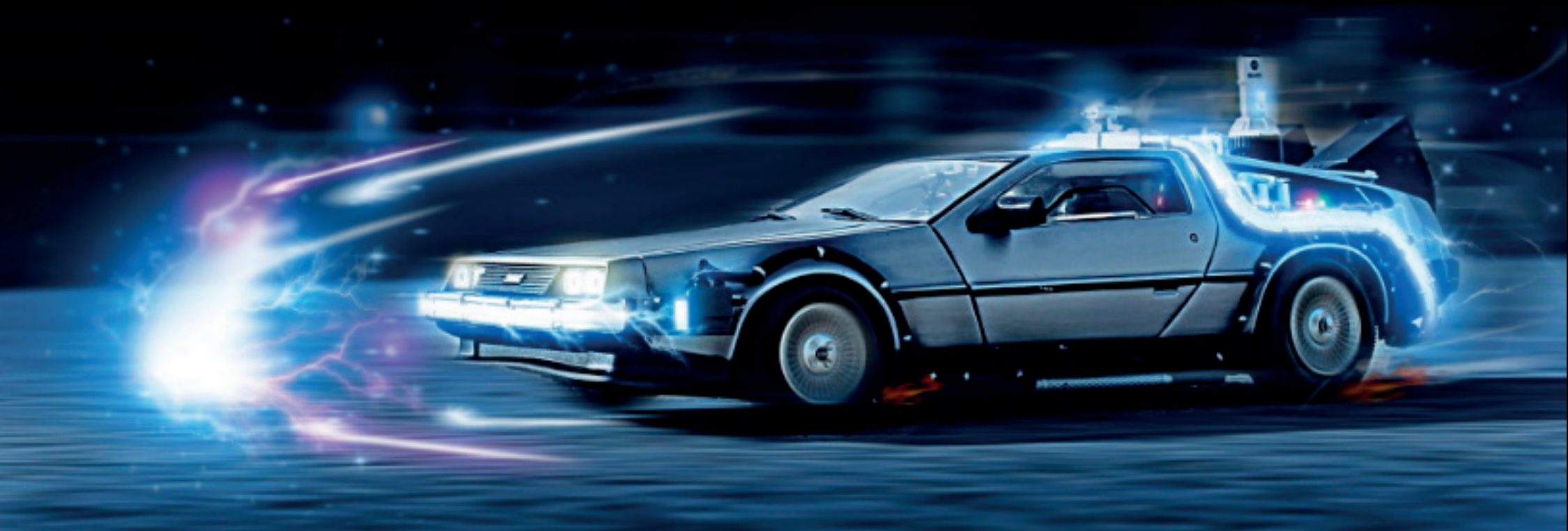
One year ago



go-reloaded

SUCCEEDED

Repo to create for this project → `https://learn.01founders.co/git/ppjd/go-reloaded`

Go	5.00 kB	go → 20% algo → 15%	1	Basic

Welcome back. Congratulations on your admission. We knew you would make it. Now it is time to get into projects.

Objectives

In this project you will use some of your old functions made in your old repository. You will use them with the objective of making a simple text completion/editing/auto-correction tool.

One more detail. This time the project will be corrected by auditors. The auditors will be other students and you will be an auditor as well.

We advise you to create your own tests for yourself and for when you will correct your auditees.

Introduction

- Your project must be written in **Go**.
- The code must respect the **good practices**.
- It is recommended to have **test files** for unit testing.

The tool you are about to build will receive as arguments the name of a file containing a text that needs some modifications (the input) and the name of the file the modified text should be placed in (the output). Next is a list of possible modifications that your program should execute:

- Every instance of `(hex)` should replace the word before with the decimal version of the word (in this case the word will always be a hexadecimal number). (Ex: "1E (hex) files were added" -> "30 files were added")
- Every instance of `(bin)` should replace the word before with the decimal version of the word (in this case the word will always be a binary number). (Ex: "It has been 10 (bin) years" -> "It has been 2 years")
- Every instance of `(up)` converts the word before with the Uppercase version of it. (Ex: "Ready, set, go (up) !" -> "Ready, set, GO !")
- Every instance of `(low)` converts the word before with the Lowercase version of it. (Ex: "I should stop SHOUTING (low)" -> "I should stop shouting")
- Every instance of `(cap)` converts the word before with the capitalized version of it. (Ex: "Welcome to the Brooklyn bridge (cap)" -> "Welcome to the Brooklyn Bridge")
    - For `(low)`, `(up)`, `(cap)` if a number appears next to it, like so: `(low, <number>)` it turns the previously specified number of words in lowercase, uppercase or capitalized accordingly. (Ex: "This is so exciting (up, 2)" -> "This is SO EXCITING")
- Every instance of the punctuations `.`, `,`, `!`, `?`, `:` and `;` should be close to the previous word and with space apart from the next one. (Ex: "I was sitting over there ,and then BAMM !!" -> "I was sitting over there, and then BAMM!!").
    - Except if there are groups of punctuation like: `...` or `!?`. In this case the program should format the text as in the following example: "I was thinking ... You were right" -> "I was thinking... You were right".
- The punctuation mark `'` will always be found with another instance of it and they should be placed to the right and left of the word in the middle of them, without any spaces. (Ex: "I am exactly how they describe me: ' awesome '" -> "I am exactly how they describe me: 'awesome'")
    - If there are more than one word between the two `' '` marks, the program should place the marks next to the corresponding words (Ex: "As Elton John said: ' I am the most well-known homosexual in the world '" -> "As Elton John said: 'I am the most well-known homosexual in the world'")
- Every instance of `a` should be turned into `an` if the next word begins with a vowel or a `h`. (Ex: "There it was. A amazing rock!" -> "There it was. An amazing rock!").

Usage

```
$ cat sample.txt
it (cap) was the best of times, it was the worst of times (up) , it was the age of wisdom, it was the age of foolishness (cap, 6) , 

$ go run . sample.txt result.txt

$cat result.txt
It was the best of times, it was the worst of TIMES, it was the age of wisdom, It Was The Age Of Foolishness, it was the epoch of be

$ cat sample.txt
Simply add 42 (hex) and 10 (bin) and you will see the result is 68.

$ go run . sample.txt result.txt

$cat result.txt
Simply add 66 and 2 and you will see the result is 68.

$ cat sample.txt
There is no greater agony than bearing a untold story inside you.

$ go run . sample.txt result.txt

$cat result.txt
There is no greater agony than bearing an untold story inside you.

$ cat sample.txt
Punctuation tests are ... kinda boring ,don't you think !?

$ go run . sample.txt result.txt

$cat result.txt
Punctuation tests are... kinda boring, don't you think!?
```

This project will help you learn about :

- The Go file system(**fs**) API
- String and numbers manipulation

> Something is wrong ? Submit an issue or even better propose a change !

This project will help you learn about :

- The Go file system(**fs**) API
- String and numbers manipulation

> Something is wrong ? Submit an issue or even better propose a change !

Audits

3 AUDITS REQUIRED

PROJECT SUCCEEDED

- ✓ gymlad
- ✓ sonalkamat
- ✓ eternal17
- ✓ arsenf
- ✓ Andru1

The code

main.go

```go
package main // editing /auto correction tool

import (
    "bufio"
    "fmt"
    "log"
    "os"
    "strconv"
    "strings"
)

func stringremove(slice []string, s int) []string { // Removes a string from a slice of string
    return append(slice[:s], slice[s+1:]...)
}

func runeremove(slice []rune, s int) []rune { // Removes a string from a slice of string
    return append(slice[:s], slice[s+1:]...)
}

func main() { // open and scan
    var strarr []string
    file, err := os.Open(os.Args[1])
    if err != nil {
        log.Fatal(err)
    }
    defer file.Close()
    fileScanner := bufio.NewScanner(file)
    fileScanner.Split(bufio.ScanWords)
    for fileScanner.Scan() {
        strarr = append(strarr, fileScanner.Text())
    }

    if err := fileScanner.Err(); err != nil {
        log.Fatal(err)
    } // calling in functions
    for i, str := range strarr {
        // fmt.Println(i, len(str), str)
        if str == "(hex)" {
            strarr[i-1] = hexaNumberToInteger(strarr[i-1])
            strarr = remove(strarr, i)
            i--
        }

        if str == "(bin)" {
            strarr[i-1] = bintodecimal(strarr[i-1])
            strarr = remove(strarr, i)
            i--
        }

        if str == "(up)" {
            strarr[i-1] = toUpper(strarr[i-1])
            strarr = remove(strarr, i)
            i--
        }

        if str == "(low)" {
            strarr[i-1] = tolower(strarr[i-1])
            strarr = remove(strarr, i)
            i--
        }

        if str == "(cap)" {
            strarr[i-1] = caps(strarr[i-1])
            strarr = remove(strarr, i)
            i--
        }

        // (remove hex/bin etc)
        if str == "(cap," {
            x := trimatoi(strarr[i+1])
            for u := 1; u ≤ x; u++ {
                strarr[i-u] = caps(strarr[i-u])
            }
            strarr = remove(strarr, i)
            i--
            strarr = remove(strarr, i+1)
            i--
        } // (remove up+number),

        if str == "(up," {
            x := trimatoi(strarr[i+1])
            for u := 1; u ≤ x; u++ {
                strarr[i-u] = toUpper(strarr[i-u])
            }
            i--
            strarr = remove(strarr, i+1)
            i--
        }

        if str == "(low," { // remove low
            x := trimatoi(strarr[i+1])
            for u := 1; u ≤ x; u++ {
                strarr[i-u] = tolower(strarr[i-u])
            }
            strarr = remove(strarr, i)
            i--
            strarr = remove(strarr, i+1)
            i--
        }
    } // output amends to result file
    strarr = punct(strarr)
    strarr = speech(strarr)
    strarr = atoan(strarr)
    resultString := strings.Join(strarr, " ")
    file, err = os.OpenFile(os.Args[2], os.O_WRONLY|os.O_CREATE, 0666)
    if err ≠ nil {
        fmt.Println("File does not exists or cannot be created")
        os.Exit(1)
    }
    defer file.Close()
    file.WriteString(resultString)
    file.Sync()
}
```

```
                if len(strarr[i]) == 1 && u == 0 {
                    strarr[i-1] += string(strRune)
                    strarr = remove(strarr, i)
                    i--
                } else if len(strarr[i]) > 1 && u == 0 && !found {
                    strarr[i-1] += string(strRune)
                    srune := []rune(strarr[i])
                    srune = runeremove(srune, u)
                    strarr[i] = string(srune)
                }
            }
        }
    }
    return strarr
}

// The punctuation mark ''' should not have spaces if there are letters in both sides of it. Otherwise, the mark should be plac

func speech(strarr []string) []string {
    quoteFound := false
    for i := 0; i < len(strarr); i++ {
        if strarr[i] == "'" {
            if !quoteFound {
                strarr[i+1] = "'" + strarr[i+1]
                strarr = remove(strarr, i)
                i--
                quoteFound = true
            } else {
                strarr[i-1] += "'"
```

```
// removing instance Every instance of (cap) converts the letter placed before into a capital.
func caps(s string) string {
    s = tolower(s)
    srune := []rune(s)
    srune[0] = srune[0] - 32
    return string(srune)
}

// replacing every instance of "a" turned into "an" if the next words begins with a vowel or an 'h'

func atoan(strarr []string) []string {
    for i, srune := range strarr {
        for _, rune := range srune {
            if (rune == 'a' || rune == 'A') && len(srune) == 1 {
                if strarr[i+1][0] == 'a' || strarr[i+1][0] == 'e' ||
                    strarr[i+1][0] == 'i' ||
                    strarr[i+1][0] == 'o' ||
                    strarr[i+1][0] == 'u' ||
                    strarr[i+1][0] == 'h' {
                    strarr[i] += "n"
                }
            }
        }
    }
    return strarr
}
```


OUTATIME