CSI 2120

Lab 2: Go

Exercise 1: Basic Methods

Define the type `dog` as follows:

```
type dog struct {
  name string
  race string
  female bool
}
```

Write a method that will rename a dog which can be used as follows:

```
func main() {
    fido := dog {"Fido", "Poodle", false }
    fido.rename("Cocotte")
}
```

Exercise 2: Go Channel Basics

Test what will be printed by the following

```
package main

import (
    "fmt"
    "time"
    "strconv"
)

var i int

func makeCakeAndSend(cs chan string) {
    i = i + 1
    cakeName := "Strawberry Cake " + strconv.Itoa(i)
    fmt.Println("Making a cake and sending ...", cakeName)
    cs <- cakeName //send a strawberry cake
}

func receiveCakeAndPack(cs chan string) {
    s := <-cs //get whatever cake is on the channel
    fmt.Println("Packing received cake: ", s)
}

func main() {
    cs := make(chan string)
    for i := 0; i<3; i++ {
        go makeCakeAndSend(cs)
        go receiveCakeAndPack(cs)

        //sleep for a while so that the program doesn't exit
        //immediately and output is clear for illustration
```

```
        time.Sleep(1 * 1e9)
    }
}
```

Exercise 3:

The following program is to be changed such that it uses arbitrary sized slices instead of fixed size arrays. In particular, the user entered size (variable sz) is to replace the hard-coded 32 in main and any function changed correspondingly.

```
package main

import "fmt"
import "math"
import "math/rand"

type Series struct {
        a, b float64
}

func (s Series) add(t, TP int) float64 {
        return s.a*math.Sin(2.0*math.Pi*float64(t)/float64(TP)) + s.b*math.Cos(2.0*math.Pi*
}

func fourier(c [32]Series, t, TP int) (res float64) {
        res = c[0].a
        for n := 1; n < 32; n++ {
                res += c[n].add(t, TP)
        }
        return
}

func main() {
        TP := 4
        sz := 1
        var res float64

        // Enter size of fourier series
        fmt.Print("Size of series (1 ... 512): ")
        // Depending on your environment you may have to remove \n in the scanf
        _, err := fmt.Scanf("%d \n", &sz)
        for err != nil || sz < 1 || sz > 512 {
                fmt.Println("Must be positive integer (1...512).")
                fmt.Println("Size of series (1 ... 512): ")
                _, err = fmt.Scanf("%d \n", &sz)
        }
        fmt.Printf("Size: %d\n", sz)

        var c [32]Series

        for t := 0; t < TP; t++ {
                for k := 0; k < 32; k++ {
                        c[k].a = rand.Float64()
                        c[k].b = rand.Float64()
                }
                res += fourier(c, t, TP)
                fmt.Printf("%f ", res)
                fmt.Println()
        }
}
```