Go Assignment - Joe Luhrman

Directions

Write a program in Go that generates n weeks of random temperature data (one temperature for each day of the week from 0-100 degrees), calculates the average temperature for each week, and then sorts the average temperatures for each week from lowest to highest (don't worry about week number labels for the averages). You should use Go routines to generate the data for each week simulataneously and then calculate the average for each week simulatneously.

The program should be able to generate, calculate the average of, and finally sort the averages of any number of weeks (n). The number of weeks can be hardcoded or a user input or whatever you want as long as it can be changed and the program still works.

The program should output the raw data for each day of each week, the unsorted averages of each week, and the sorted averages of each week.

It's okay for the temperatures to be ints but obviously the averages will need to be floats. You also do not need to write your own sort (use "sort" package).

Hints

I would strongly recommend using VS Code with the Go extension (you also need to download the current version of Go from the official site). I would recommend you use slices to store the averages of each week because Go has fast built-in slice sorts.

Submission

Just a .go file with your code.

```go
/*
 Joe Luhrman
 Dr. Binkley
 CS 451
 Pet Language Project Assignment Solution
*/

package main

import (
 "fmt"
 "math"
 "math/rand"
 "sort"
 "time"
)

const (
 DAYS = 7
 N    = 4 // number of weeks to use
)

func main() {

 var week_data [N][DAYS]int      // store temps for each day (simpler to just use array instead of slice)
 week_avgs := make([]float64, N) // store avgs for each week (used slice for easier sorting later)

 // ensures random numbers arent the same every time you run
 rand.Seed(time.Now().UnixNano())

 // generating temperatures for each day
 for i := 0; i < N; i++ {
  data := make(chan [DAYS]int)
  go func() { data <- generateDays() }() // routine to generate the data for each week simultaneously (uses anonymous function)
  week_data[i] = <-data                  // receive the data from the channel when ready
 }

 // calculating week avgs
 for i := 0; i < N; i++ {
  avg := make(chan float64)
  go func() { avg <- calculateAvg(week_data[i]) }()
  week_avgs[i] = <-avg
 }

 // unsorted data
 fmt.Println("\nData: ")
 printWeekData(week_data)
 fmt.Println("\nAverages (unsorted): ")
 printWeekAvgs(week_avgs) // in order of week number

 // sorting
 sort.Float64s(week_avgs)

 // final result
 fmt.Println("\nAverages (sorted): ")
 printWeekAvgs(week_avgs)

}

/*
 calculates the average of an array of int values (temperatures for each day)
 param  -> days - the array of ints
 return -> avg - the float64 average of the ints
*/
func calculateAvg(days [DAYS]int) float64 {
 var avg float64
 sum := 0
 for i := 0; i < DAYS; i++ {
  sum += days[i]
 }
 avg = math.Round((float64(sum)/float64(DAYS))*100) / 100 // rounds to nearest 2 decimal places

 return avg
}

/*
 generates random int temperatures for every day of a week
 return -> days - the array of temps between 0-100
*/
func generateDays() [DAYS]int {
 var days [DAYS]int

 for i := 0; i < DAYS; i++ {
  days[i] = rand.Intn(101)
 }

 return days
}

/*
 prints out the raw temperature data for each week
 param  -> week_data - the temperature data for each day of each week
*/
func printWeekData(week_data [N][DAYS]int) {
 for i := 0; i < N; i++ {
  fmt.Print("Week ", i, ":")
  for j := 0; j < DAYS; j++ {
   fmt.Print(" ", week_data[i][j])
  }
  fmt.Print("\n")
 }
}

/*
 prints the average temperatures for each week
 param -> week_avgs - the average temperatures for each week (float64)
*/
func printWeekAvgs(week_avgs []float64) {
 for i := 0; i < N; i++ {
  fmt.Println("Week ", i, ": ", week_avgs[i])
 }
}
```