review

we’ve covered a lot!

exercises

gravatar

- Create a program which
  - you run from the command line
  - reads user information (at least name and email) from the command line (os.Args[1])
  - creates an HTML page
  - generates a profile image from gravatar.com
    - gravatar hash
    - gravatar image
  - part of your solution should include this code:

```
h := md5.New()
io.WriteString(h, email)
finalBytes := h.Sum(nil)
finalString := hex.EncodeToString(finalBytes)
```

word count

- create a program that
  - returns a map of the counts of each “word” in a string. (strings.Fields)
  - WordCount(“test test”) → map[string]int{ “test”: 2 }

centered average

- create a program that
  - computes the average of a list of numbers, but removes the largest and smallest values.
  - centeredAverage([]float64{1, 2, 3, 4, 100}) → 3

swap

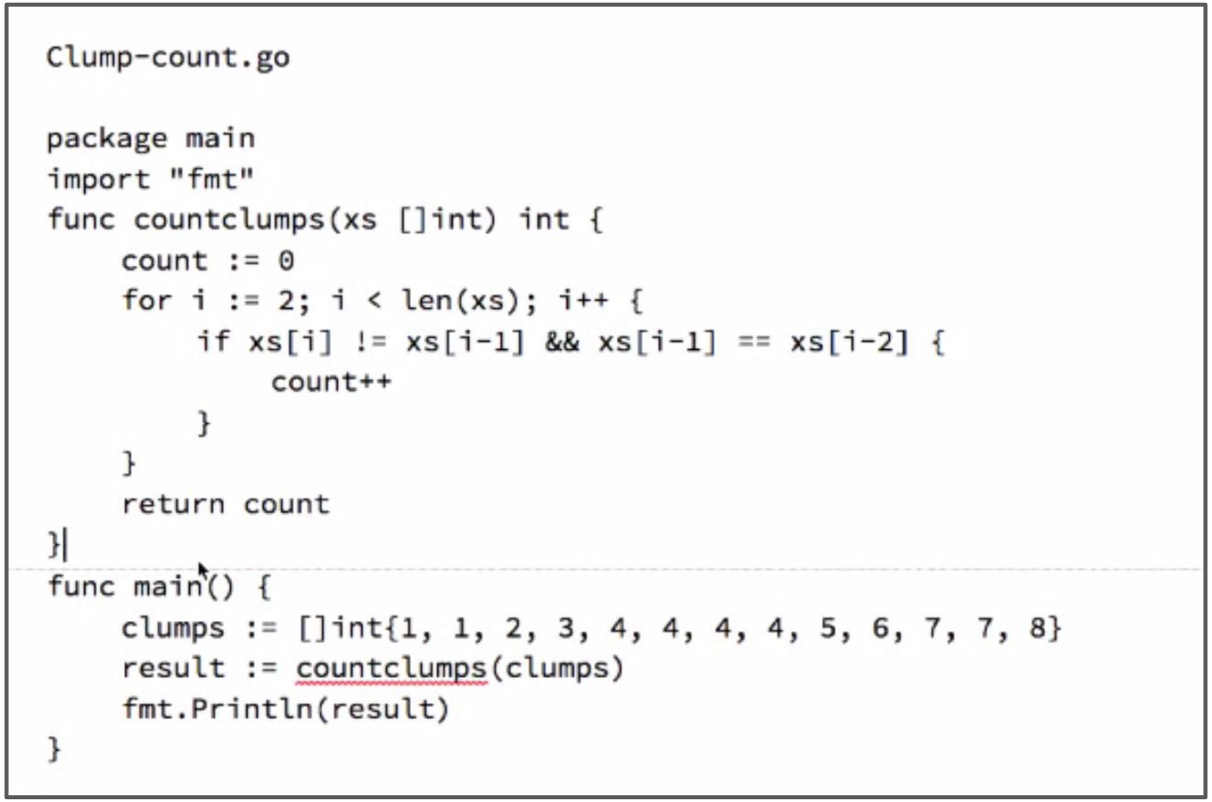
Write a program that can swap two integers

```
x := 1
y := 2
swap(&x, &y)
```

should give you x=2 and y=1

clumps

- Say that a "clump" in a list is a series of 2 or more adjacent elements of the same value.
  - Return the number of clumps in the given list.
    - countClumps([]int{1, 2, 2, 3, 4, 4}) → 2
    - countClumps([]int{1, 1, 2, 1, 1}) → 2
    - countClumps([]int{1, 1, 1, 1, 1}) → 1



```
Clump-count.go

package main
import "fmt"
func countclumps(xs []int) int {
	count := 0
	for i := 2; i < len(xs); i++ {
		if xs[i] != xs[i-1] && xs[i-1] == xs[i-2] {
			count++
		}
	}
	return count
}
func main() {
	clumps := []int{1, 1, 2, 3, 4, 4, 4, 4, 5, 6, 7, 7, 8}
	result := countclumps(clumps)
	fmt.Println(result)
}
```

one attempt; has an error though

cat <file>

Create your own version of cat which reads a file and dumps it to stdout.

copy <file>

Create a program which opens a file, reads a file, then writes the contents to a new file.

cp <src file> <dst file>

Create your own version of cp which reads a file and writes it to another file.

unnamed field

Why would you use an embedded anonymous unnamed field instead of a normal named field?

sentence case

Create a program which converts the first character of each line in a file to uppercase and writes it to stdout.

capitalize the first letter of every word

Create a program which capitalizes the first letter of every word from a text file and writes it to stdout.

capitalize every other word

Create a program which capitalizes every other word (capitalizes the entire word) from a text file and writes it to stdout.

count words

Count how many times the word “whale” is used in Herman Melville’s novel, “Moby Dick” (927 pages).

Download “Moby Dick” for free from here:
http://www.gutenberg.org/files/2701/old/moby10b.txt

At terminal:

```
curl -O http://www.gutenberg.org/files/2701/old/moby10b.txt
```

longest word

Find the longest word (string of runes not separated by spaces) used in “Moby Dick”.

Download “Moby Dick” for free from here:
http://www.gutenberg.org/files/2701/old/moby10b.txt

At terminal:

```
curl -O http://www.gutenberg.org/files/2701/old/moby10b.txt
```

cat <files>

Create your own version of cat which reads an unlimited number of file and dumps their contents to stdout.

csv state info

- Download a CSV file of state information from statetable.com
- implement a program which
  - parses all of the csv file
  - prints the following fields for each state:

id	int
name	string
abbreviation	string
censusRegionName	string

  - Make sure to use a struct in your program

csv state info - lookup by state

- Download a CSV file of state information from statetable.com
- implement a program which
  - parses all of the csv file
  - prints the following fields for each state:

id	int
name	string
abbreviation	string
censusRegionName	string

  - Make sure to use a struct in your program
- accept a state abbreviation from the user
  - display only the requested state

csv state info - lookup by state - display html table

- Download a CSV file of state information from statetable.com
- implement a program which
  - parses all of the csv file
  - prints the following fields for each state:

id	int
name	string
abbreviation	string
censusRegionName	string

  - Make sure to use a struct in your program
- accept a state abbreviation from the user
  - write the requested state into an html file

csv stock price

- Grab historical financial data from Yahoo as a csv file
    - http://finance.yahoo.com/q/hp?s=GOOG+Historical+Prices)
- read that file
- print content from each record to standard out

csv stock price

- Grab historical financial data from Yahoo as a csv file
  - http://finance.yahoo.com/q/hp?s=GOOG+Historical+Prices)
- read that file
- print content from each record to an html table

checksum a file

- Create a program which finds the md5 checksum of a file.
    - at the terminal
        - go install
        - yourProgramName yourFileNameToCheckSum
            - should return a large hexadecimal number like ‘d47c2bbc28298ca9befdfbc5d3aa4e65’

checksum a directory of files

- Create a program which finds the md5 checksum of all of the files in a directory.