CIS 424 Sec 50

Fall 2018

Project 6

Emily Ferline & Victor Ipinmoroti

ID: viipinmo

PROJECT 6

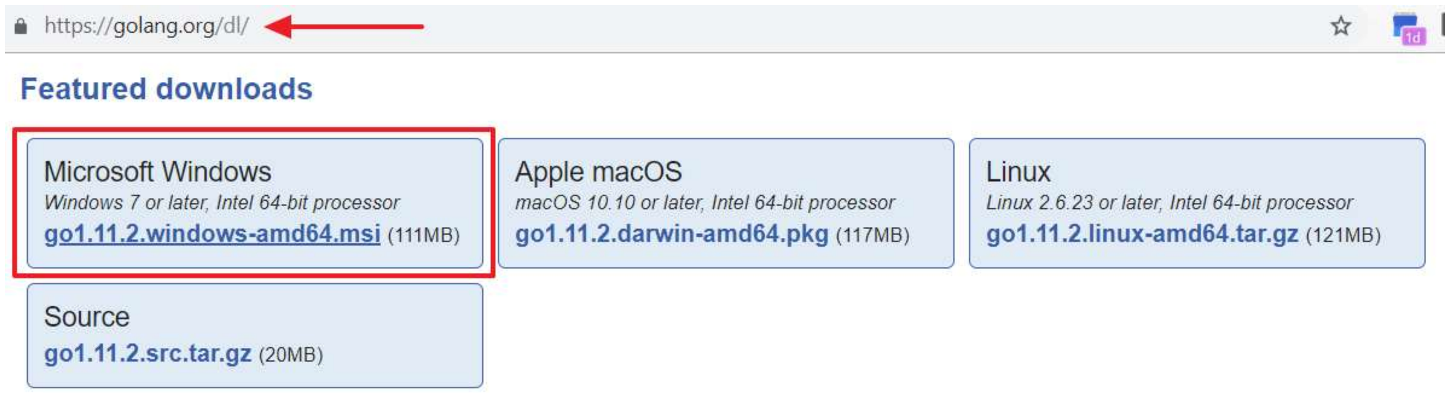
TABLE OF CONTENTS

INTRODUCING GO LANGUAGE

WHAT IS GO

Go (aka 'Golang') is open source language, built by Google, and intended for projects that are scalable like systems languages, web development, etc. This language is also static typed, like C++, which enhances speed. There is no need for the program to search around for variable types which may have changed (integers, floats, etc.), the developer is able to control this and tell the program exactly what to look for.

The installation files for Go are located at https://golang.org/dl/ and can be installed for Windows, Mac, or Linux. For my purposes, the .msi file for Windows download were needed.


https://golang.org/dl/

Featured downloads

Microsoft Windows
Windows 7 or later, Intel 64-bit processor
go1.11.2.windows-amd64.msi (111MB)

Apple macOS
macOS 10.10 or later, Intel 64-bit processor
go1.11.2.darwin-amd64.pkg (117MB)

Linux
Linux 2.6.23 or later, Intel 64-bit processor
go1.11.2.linux-amd64.tar.gz (121MB)

Source
go1.11.2.src.tar.gz (20MB)

The syntax is configured so that declaration and initialization of variables doesn't require the type to be specified. For example, **`x := 30`** in Golang is the same as **`int x = 30`** in C language. Regarding types, there are numerous numeric, Boolean and character types. Interestingly enough, there are two features in Go that are used instead of class inheritance: *embedding* – happens when an anonymous field in a struct is included… and *interfaces* – which are named collections of method signatures.

From a debugging perspective, Go has many tools that can help. Delve is referenced on the golang.org website as a better alternative when debugging using standard toolchain [1].

IMPORTANCE OF HIGHER ORDER FUNCTIONS

Higher order functions are functions that accept another function as an argument, referred to as a "callback", and then returns a function as a result. The three most common built-in higher order functions are called *map*, *filter*, and *reduce*.

A large advantage to using higher order functions is for program organization and streamlining. Higher order functions are essentially commonly used patterns of design that are built into functional forms. Map, filter, and reduce can be used in replacement of many predefined functions in programming languages to perform the same task.

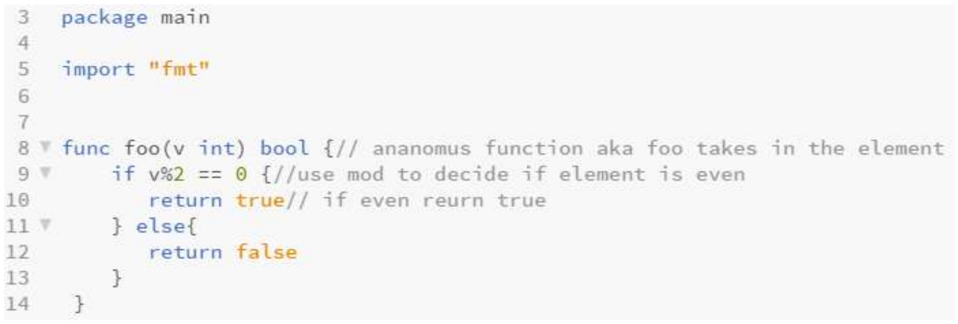
OTHER ADVANTAGES

Other than increasing overall program modularity and functionality with higher order functions, they also provide other benefits, such as recursion. Additionally, they can be used together or separately. Knowing the best time to use a higher order function also proves its own benefits. According to quora.com, if you are constructing a program that takes a Boolean or number in need of customizing its behavior, a higher order function would be of more benefit to use. [2]

HIGHER ORDER FUNCTION PROGRAM + EXPLANATION OF CODE

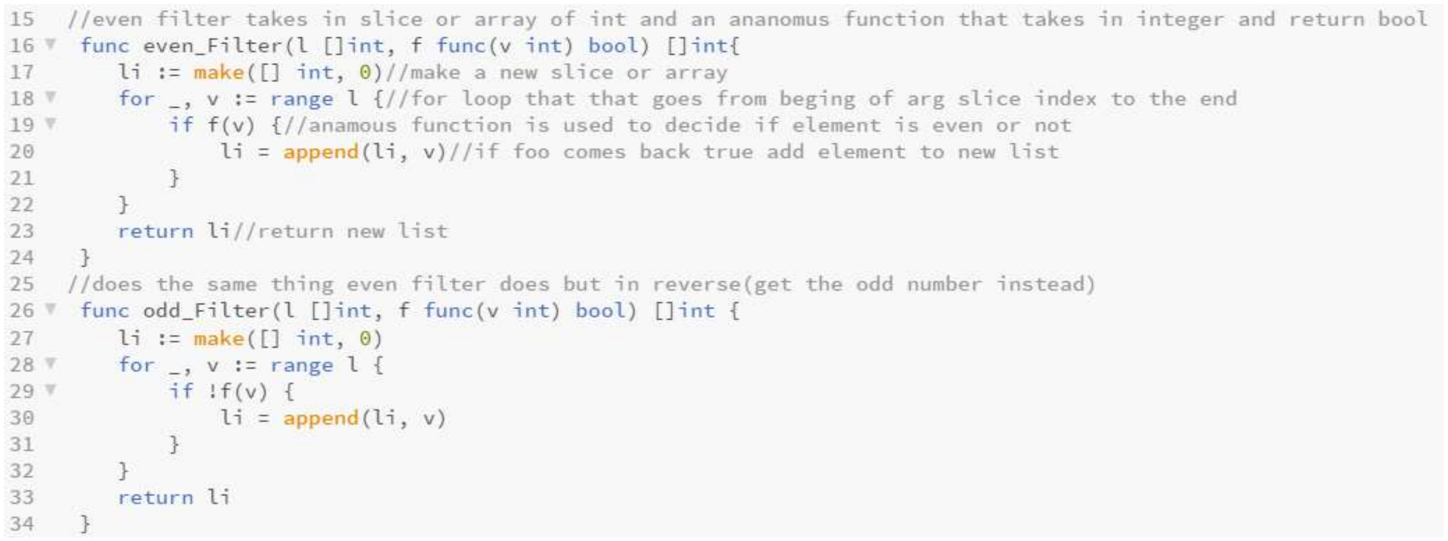
Our higher order function program retrieves a predefined list of numeric elements from the user and calculates the minimum, maximum, and sum of even or odd (or both) numbers using *filter* and *reduce*.

The program starts by importing '`fmt`' used for input/output functions. The first function defined is an anonymous function that evaluates the elements to determine if it's even or odd – returning true for even, and false for odd.



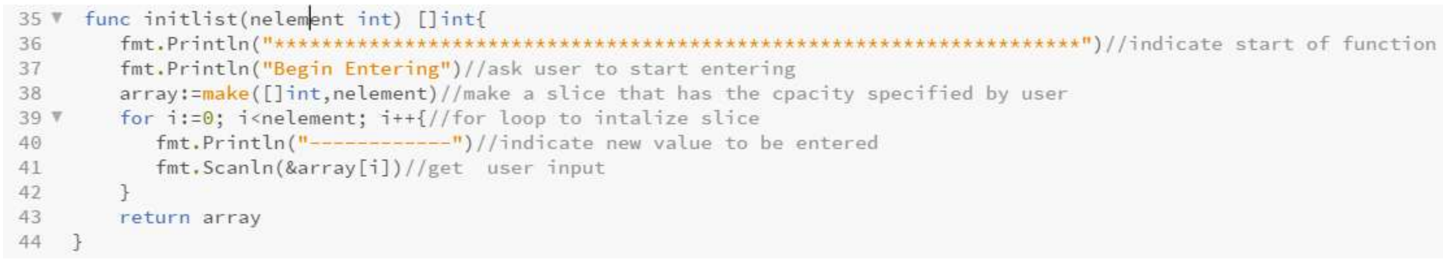
```
package main

import "fmt"


func foo(v int) bool {// ananomus function aka foo takes in the element
    if v%2 == 0 {//use mod to decide if element is even
        return true// if even reurn true
    } else{
        return false
    }
 }
```

Two more functions are defined to get even or odd numbers, called `even_Filter` and `odd_Filter`. *These functions get the anonymous function foo as an argument and use it to determine to filter the list based on the bolean state of foo. In even_filter we want foo to be true in odd we want foo to be false .*



```
//even filter takes in slice or array of int and an ananomus function that takes in integer and return bool
 func even_Filter(l []int, f func(v int) bool) []int{
    li := make([] int, 0)//make a new slice or array
    for _, v := range l {//for loop that that goes from beging of arg slice index to the end
        if f(v) {//anamous function is used to decide if element is even or not
            li = append(li, v)//if foo comes back true add element to new list
        }
    }
    return li//return new list
 }
//does the same thing even filter does but in reverse(get the odd number instead)
 func odd_Filter(l []int, f func(v int) bool) []int {
    li := make([] int, 0)
    for _, v := range l {
        if !f(v) {
            li = append(li, v)
        }
    }
    return li
 }
```

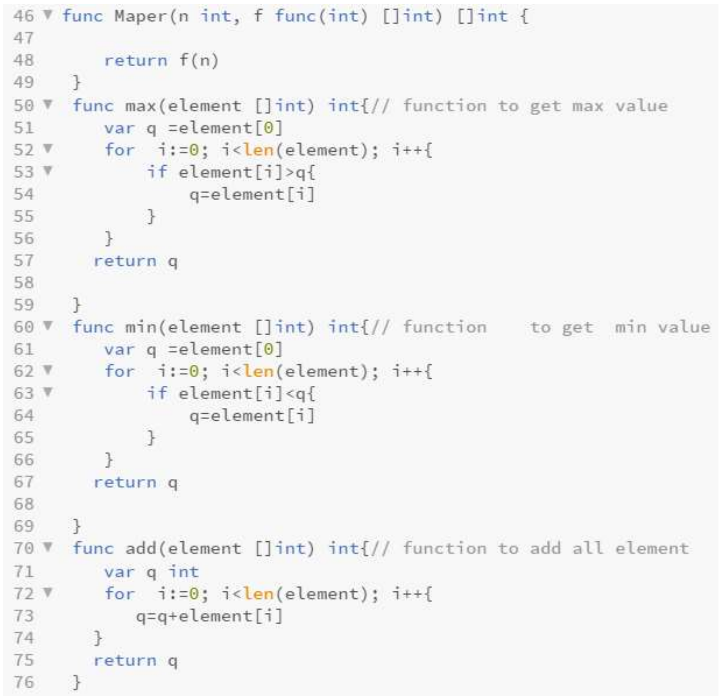
Function `initlist()` is defined to prompt the user for the capacity of the array list to be manipulated. Slices in Golang have both a length and a capacity. [3] The capacity is the total amount of elements in the underlying array. *The function is technically an anonymous function accepted by mapper and is returned by mapper.*



```go
func initlist(nelement int) []int{
   fmt.Println("********************************************************************")//indicate start of function
   fmt.Println("Begin Entering")//ask user to start entering
   array:=make([]int,nelement)//make a slice that has the cpacity specified by user
   for i:=0; i<nelement; i++{//for loop to intalize slice
      fmt.Println("------------")//indicate new value to be entered
      fmt.Scanln(&array[i])//get  user input
   }
   return array
}
```

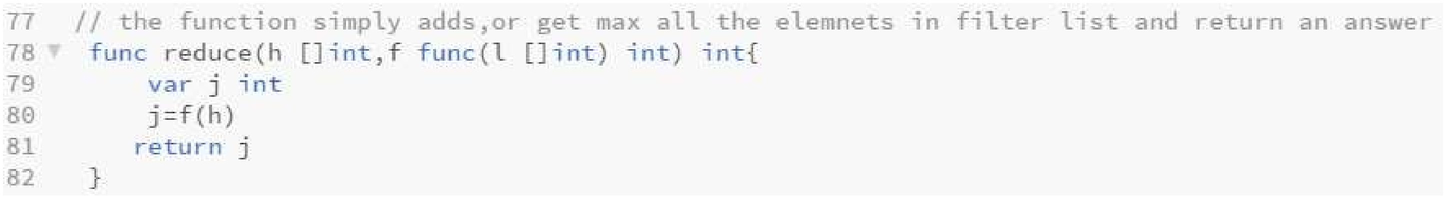
Three more functions were defined to return the maximum element, minimum element, and summation of all elements called `max`, `min`, and `add`. Keyword `len` in Golang is used to identify the length. A for loop is used to traverse all the elements in the list and update the return result with the maximum, minimum, and summation.

As stated before the mapper gets the list from the function initlisit() and returns it in the main program, if we wanted to change how the list is ordered or gets inputted we can pass in a different function to do it . For the purpose of this program we do not need to do that we needed to just get a list from the user but it show one of the advantage of using higher order function. It improves the readability of the code.



```go
func Maper(n int, f func(int) []int) []int {

    return f(n)
  }
  func max(element []int) int{// function to get max value
    var q =element[0]
    for  i:=0; i<len(element); i++{
        if element[i]>q{
            q=element[i]
        }
    }
   return q

  }
  func min(element []int) int{// function    to get  min value
    var q =element[0]
    for  i:=0; i<len(element); i++{
        if element[i]<q{
            q=element[i]
        }
    }
   return q

  }
  func add(element []int) int{// function to add all element
    var q int
    for  i:=0; i<len(element); i++{
       q=q+element[i]
   }
   return q
  }
```

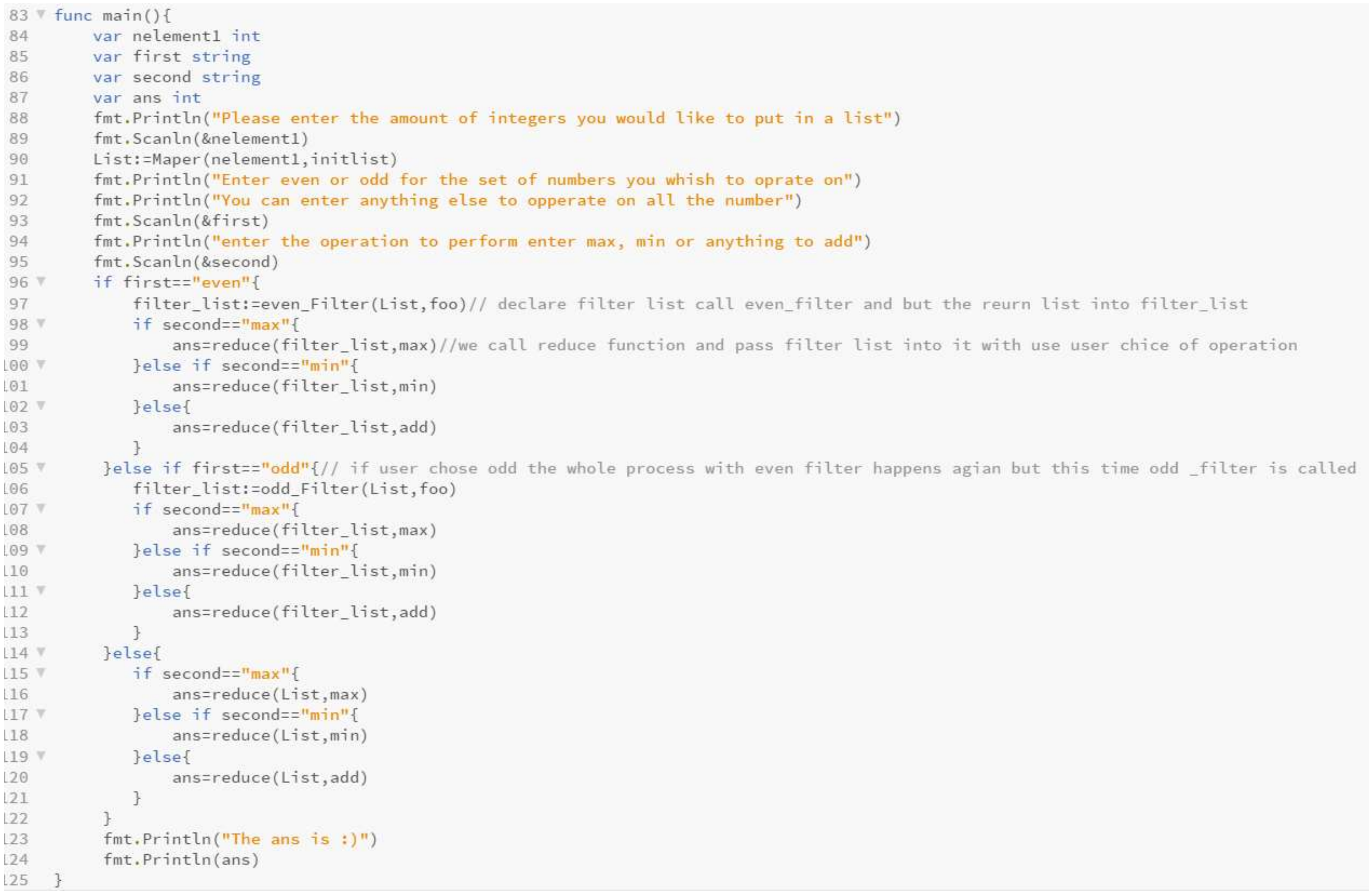
The reduce function is used to add or get the max of all elements and return the result. This higher order function takes functions as parameters and returns a result.



```
// the function simply adds,or get max all the elemnets in filter list and return an answer
 func reduce(h []int,f func(l []int) int) int{
     var j int
     j=f(h)
    return j
 }
```

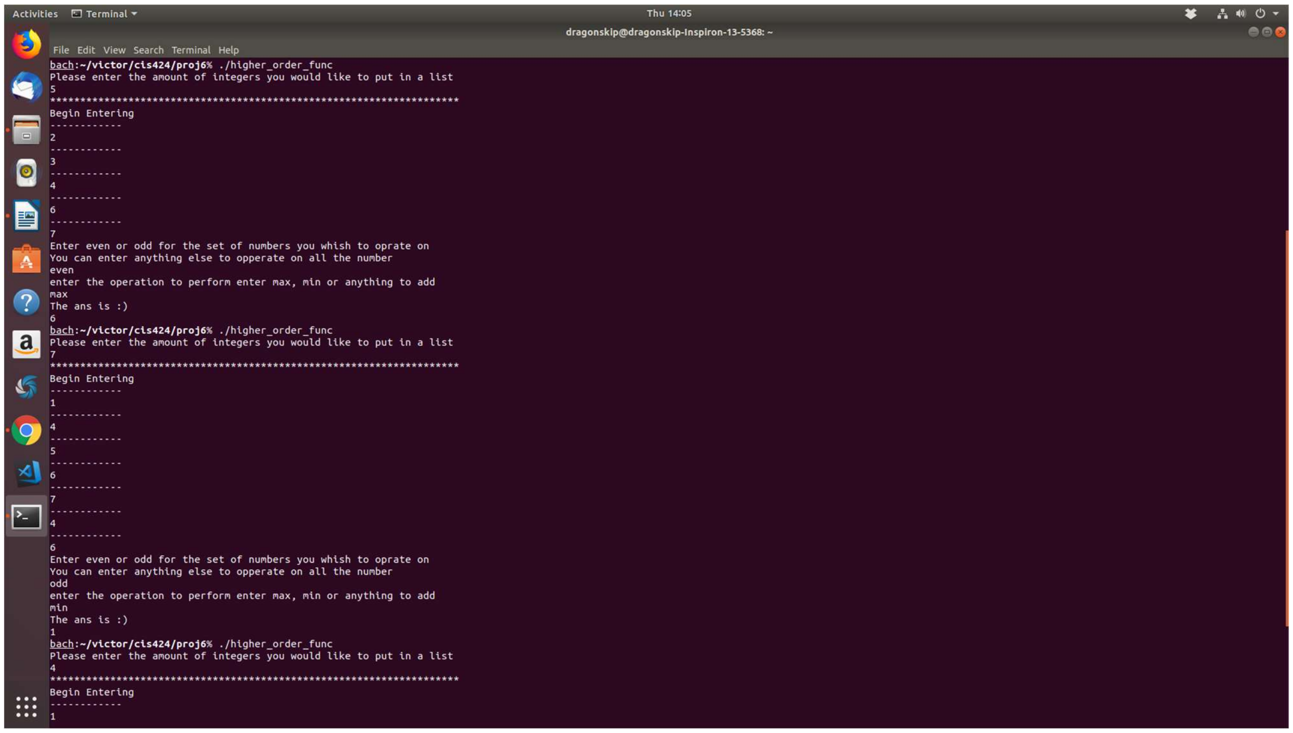
And lastly, the `main()` function is shown below. Four variables are declared `nelement1`, `first`, `second`, and `ans`. `nelement1` as integer is used to hold the value from the user for how long the array should be. The variable `first` is used to identify from the user either even or odd as input. `second` will hold the value for which operation to be performed on the list, `max`, `min`, or `add`.

If-else statements are then used to evaluate the user input and perform the required operations. The first if statement determines if the user input matches “even,” and if it’s true – `even_Filter` function is called and the list is returned to `filter_list`. The second if statement determines the operation that was entered, max, min, or add, using else-if to traverse the options. From there, the corresponding function is passed to `filter_list`. The final segment of code will display the answer.

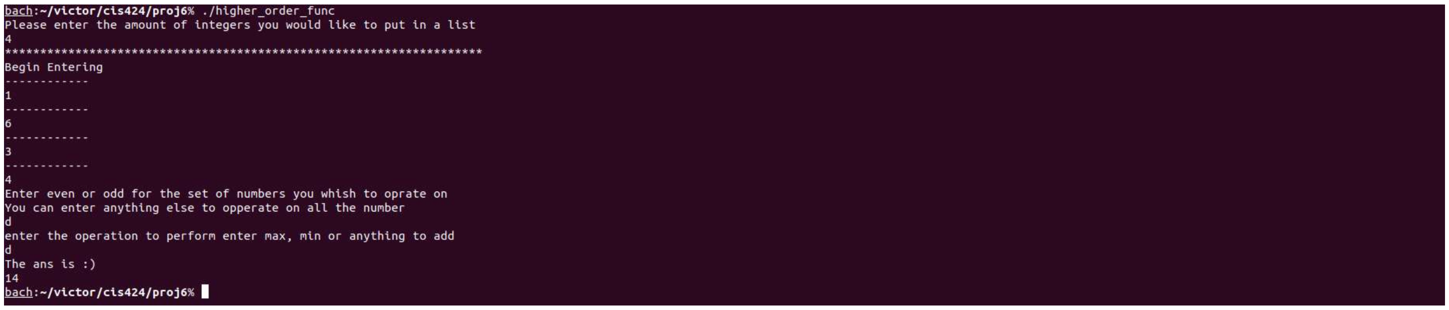


```
func main(){
    var nelement1 int
    var first string
    var second string
    var ans int
    fmt.Println("Please enter the amount of integers you would like to put in a list")
    fmt.Scanln(&nelement1)
    List:=Maper(nelement1,initlist)
    fmt.Println("Enter even or odd for the set of numbers you whish to oprate on")
    fmt.Println("You can enter anything else to opperate on all the number")
    fmt.Scanln(&first)
    fmt.Println("enter the operation to perform enter max, min or anything to add")
    fmt.Scanln(&second)
    if first=="even"{
        filter_list:=even_Filter(List,foo)// declare filter list call even_filter and but the reurn list into filter_list
        if second=="max"{
            ans=reduce(filter_list,max)//we call reduce function and pass filter list into it with use user chice of operation
        }else if second=="min"{
            ans=reduce(filter_list,min)
        }else{
            ans=reduce(filter_list,add)
        }
     }else if first=="odd"{// if user chose odd the whole process with even filter happens agian but this time odd _filter is called
        filter_list:=odd_Filter(List,foo)
        if second=="max"{
            ans=reduce(filter_list,max)
        }else if second=="min"{
            ans=reduce(filter_list,min)
        }else{
            ans=reduce(filter_list,add)
        }
     }else{
        if second=="max"{
            ans=reduce(List,max)
        }else if second=="min"{
            ans=reduce(List,min)
        }else{
            ans=reduce(List,add)
        }
     }
     fmt.Println("The ans is :)")
     fmt.Println(ans)
}
```

Program output



```
bach:~/victor/cis424/proj6% ./higher_order_func
Please enter the amount of integers you would like to put in a list
5
*******************************************************************
Begin Entering
-----------
2
-----------
3
-----------
4
-----------
6
-----------
7
Enter even or odd for the set of numbers you whish to oprate on
You can enter anything else to opperate on all the number
even
enter the operation to perform enter max, min or anything to add
max
The ans is :)
6
bach:~/victor/cis424/proj6% ./higher_order_func
Please enter the amount of integers you would like to put in a list
7
*******************************************************************
Begin Entering
-----------
1
-----------
4
-----------
5
-----------
6
-----------
7
-----------
4
-----------
6
Enter even or odd for the set of numbers you whish to oprate on
You can enter anything else to opperate on all the number
odd
enter the operation to perform enter max, min or anything to add
min
The ans is :)
1
bach:~/victor/cis424/proj6% ./higher_order_func
Please enter the amount of integers you would like to put in a list
4
*******************************************************************
Begin Entering
-----------
1
```



```
bach:~/victor/cis424/proj6% ./higher_order_func
Please enter the amount of integers you would like to put in a list
4
*******************************************************************
Begin Entering
-----------
1
-----------
6
-----------
3
-----------
4
Enter even or odd for the set of numbers you whish to oprate on
You can enter anything else to opperate on all the number
d
enter the operation to perform enter max, min or anything to add
d
The ans is :)
14
bach:~/victor/cis424/proj6%
```

Biographies

Emily Ferline – IT Systems Administrator, NeoGraf Solutions LLC.

I am a BSCIS major about to start my final semester at Cleveland State in the Spring of 2019, and I plan to graduate in May 2019. I have worked in the IT field for 8 years for GrafTech International Holdings, Inc. (2010 – 2016) and NeoGraf Solutions, LLC (2017 - present). I started as an intern in 2010 where I learned the fundamentals of desktop support related tasks, networking, and light server administration. After a 2-year internship, I was hired as a Desktop Support Analyst for GrafTech and continued learning and growing on the infrastructure side of IT for 5 more years. I have recently taken the role of IT Systems Administrator for NeoGraf Solutions where I support an infrastructure of over 40 virtual machines, 7 physical servers, and over 10 Cisco network devices.

My personal creative outlet (and hobby) is baking gourmet cakes for friends, family, and anyone who needs a fancy dessert for any occasion! I have built a website to showcase my designs and continually update my social media platform with new creations. I also enjoy Latin ballroom dancing and occasionally competing in local DanceSport events in Cleveland area.

Victor Ipinmoroti – CSU Student

I am a computer science student in my junior year. I was originally born in Nigeria until I came to Massachusetts 6 years ago. I have been in Cleveland for approximately 3 years now. It was around this time I graduated high school and joined Cleveland state university. I plan to go into web app development when I graduate college. I am a curios person by nature and like to learn new things even though I might be terrible at it. I first took interest in programing when I was 15 that was 2 years after I came to the united states. It was more of a hobby for me back then as I was more interested in physics at the time. As a matter of fact, I thought was going to study physics or mathematics in college back then. Cut to 3 years later and I decided to study computer science because I saw the practicality of it. This is not saying mathematics was not practical enough for me I just thought computer science might be more hands-on.

REFERENCES

[1] https://golang.org/doc/gdb - Debugging Go Code

[2] https://www.quora.com/When-should-I-use-higher-order-functions - Introduction

[3] https://tour.golang.org/moretypes/11 - Slices - Capacity and Length

[4]https://gobyexample.com/collection-functions -