```swift
//
//  main.swift
//  End of Year Project
//
//  Created by Eddie Ceausu on 12/12/16.
//  Copyright © 2016 Eddie Ceausu. All rights reserved.
//

// ********************************************************************
// Imports
// ********************************************************************
import Foundation
import Swift
import Darwin
import Quartz
//
*********************************************************************
// Global Variables
//
*********************************************************************
var done = false
var done3 = false
var arraysize: Int
var upperlimit: UInt32
var response: String
//
*********************************************************************
//
// Function strinput returns a String which it reads from the Console
//
//
*********************************************************************
func strinput() -> String {
    let keyboard = FileHandle.standardInput
    let inputData = keyboard.availableData
    let strData = NSString(data: inputData, encoding:
String.Encoding.utf8.rawValue)!

    return strData.trimmingCharacters(in: CharacterSet.newlines)
} // end of function strinput
//
*********************************************************************
//
// Function returns an integer which it reads from the Console
//
//
*********************************************************************
func intinput() -> Int {
    let keyboard = FileHandle.standardInput
    let inputData = keyboard.availableData
```

```
    let strData = NSString(data: inputData, encoding:
String.Encoding.utf8.rawValue)!
    strData.trimmingCharacters(in: CharacterSet.newlines)

    return Int (strData.intValue)
} // end of function intinput
//
*************************************************************************
func Uintinput() -> UInt32 {
    let keyboard = FileHandle.standardInput
    let inputData = keyboard.availableData
    let strData = NSString(data: inputData, encoding:
String.Encoding.utf8.rawValue)!
    strData.trimmingCharacters(in: CharacterSet.newlines)

    return UInt32 (strData.intValue)
} // end of function Uintinput
//
*************************************************************************
//
// Function returns a Double which it reads from the Console
//
//
*************************************************************************
func doubleinput() -> Double {
    let keyboard = FileHandle.standardInput
    let inputData = keyboard.availableData
    let strData = NSString(data: inputData, encoding:
String.Encoding.utf8.rawValue)!
    strData.trimmingCharacters(in: CharacterSet.newlines)

    var dvalue: Double = 0

    dvalue = strData.doubleValue

    return dvalue
} // end of function doubleinput
//
*************************************************************************
//
// Function readfile reads a file into a large array of strings
// each item in the array is a line from the file
//
//
*************************************************************************
func readfile(_ path:String) -> [String]{
    do {
        // Read an entire text file into an NSString.
        let contents = try NSString(contentsOfFile: path, encoding:
String.Encoding.ascii.rawValue)
```

```
        let lines:[String] = contents.components(separatedBy: ",")

        return lines
    } catch {
        print("Unable to read file: \(path)");
        return [String]() }
}
//*****************************************************************
// Functions
//
//*****************************************************************
func CustomSet() -> (set:[Double], arraysize:Int, upperlimit: UInt32)
{
    var set: [Double]
    var j: Int
    print("How many numbers are in your data set: ", terminator: "")
    let arraysize = intinput(); set = [Double] (repeating: 0, count:
arraysize)

    print("Please enter your data:", terminator:"")

    for i in 0..<set.count {
     set[i] = doubleinput()
    }

    if set.count > 10000 {
     print("\r Sotring...")
    }
    set.sort()
    j = set.count - 1
    var upperlimit = UInt32(set[j])

    return(set, arraysize, upperlimit)
}
func RandomSet() -> (set:[Double], arraysize:Int, upperlimit: UInt32)
{
    var set: [Double]
    var upperlimit: UInt32
    var start, end: Double
    var path: URL
    let file: String
    let contents: String


    print("How many numbers are in your data set: ", terminator: "")
    let arraysize = intinput()

    print("Enter the Upperlimit of your random set: ", terminator: "")
    upperlimit = Uintinput()
```

```
    set = [Double] (repeating: 0, count: arraysize)
    if arraysize >= 1000000 {
     print("\r Building...")
    }

    for i in 0 ..< arraysize { // loop will create a random array by
selecting a random uint using arc4random_uniform func
      set[i] = Double(arc4random_uniform(upperlimit))

    }

    if set.count > 100000 {
     print("\r Sotring...")
     print(" ")
    }
    start = CACurrentMediaTime()
    set.sort() // built in swift sort function
    end = CACurrentMediaTime()
    print("Time to sort with .sortInPlace is \(round((end - start) *
1000) / 1000) sec") // timing check
    if set.count > 1000 { // file creation of array is too large to
print
      file = "random_data.txt"
      contents = String(describing: set)
      if let directory =
NSSearchPathForDirectoriesInDomains(FileManager.SearchPathDirectory.do
cumentDirectory, FileManager.SearchPathDomainMask.allDomainsMask,
true).first {
           path = URL(fileURLWithPath:
directory).appendingPathComponent(file)
           print("Due to your file being larger than 1000 items, it
was placed at: \(path)")
           //writing
           do {
                try contents.write(to: path, atomically: false,
encoding: String.Encoding.utf8)
           }
           catch {print("Unable to write to: \(path)")}
     }
    }

    if set.count <= 1000 {

    print("Would you like to print your random set? (Y/N) ")
    let response = stringinput().lowercased()

     if response == "y" {
           print(" ")
           print(set)
     }
```

```
    }
    return (set, arraysize, upperlimit)
}
func StandDev2_Variance(_ set: [Double]) {
    let average: Double
    var newaverage: Double
    var final, final2: Double

    average = set.reduce(0, +) / Double(set.count)
    newaverage = set.map{pow($0 - average, 2.0)}.reduce(0, +)
    final2 = newaverage / Double(set.count)
    final = sqrt(final2)

    print("Variance: \(round(final2 * 1000) / 1000)")
    print("Standard Deviation: \(round(final * 1000) / 1000)")
}
func MinandMax(_ set: [Double]) {
    let i = 0; let minnr = set[i]
    let j = set.count - 1; let maxnr = set[j]

    print("Max value: \(maxnr)")
    print("Min value: \(minnr)")
    print("Range: \(maxnr - minnr)")
}
func average(_ set: [Double]) {
    var average: Double
    var sum: Double
    average = set.reduce(0, +) / Double(set.count)
    sum = set.reduce(0, +)
    print("Sum: \(sum)")
    print("Average/Mean: \(round(average * 1000) / 1000) ")
}
func Median(_ set: [Double]) {
    var i, j: Int
    var m: Double

    if set.count % 2 != 0 {

     i = set.count / 2
     print("Median: \(set[i])")
    }

    if set.count % 2 == 0 {

     i = set.count / 2
     j = (set.count / 2) - 1
     m = set[i] + set[j]; m = m / 2
     print("Median: \(m)")
    }
}
```

```
func Mode(_ set: [Double], arraysize: Int, Upperlimit: UInt32) {
    if Int(Upperlimit) < arraysize {
     print("Mode will not be calculated")
     return
    }

    var path: URL
    let file: String
    let contents: String
    var set3 = [Double]()
    var j, k: Double
    var set2 = set
    var index = set.count
    set2.insert(0.0, at: index)

    for i in 0...set.count - 1 {
     j = set2[i]
     k = set2[i + 1]

     if j == k {
          set3.append(j)

     }
    }

    if set3.count > 2 {
     var passes: Int = 1
     j = 0; k = 0
     set2.removeAll()
     index = set3.count
     set3.insert(0.0, at: index)
     let count = set3.count - 2

     for i in 0...count {
          passes += 1
          j = set3[i]
          k = set3[i + 1]

          if j == k {
               set2.append(j)

          }
     }
    }
    // Return statement if array is to large to display
    if set2.count > 50 {
     file = "mode_data.txt"
     contents = String(describing: set2)
     if let directory =
NSSearchPathForDirectoriesInDomains(FileManager.SearchPathDirectory.do
```

```
cumentDirectory, FileManager.SearchPathDomainMask.allDomainsMask,
true).first {
            path = URL(fileURLWithPath:
directory).appendingPathComponent(file)
            //writing
            do {
                  try contents.write(to: path, atomically: false,
encoding: String.Encoding.utf8)
            }
            catch {print("Unable to write to: \(path)")}
      }

      print("Mode is greater than 50 numbers and your file is located
in Documents directory as: \(file)")
      return

    }
    if set3.count > 50 {
      file = "mode_data.txt"
      contents = String(describing: set3)
      if let directory =
NSSearchPathForDirectoriesInDomains(FileManager.SearchPathDirectory.do
cumentDirectory, FileManager.SearchPathDomainMask.allDomainsMask,
true).first {
            path = URL(fileURLWithPath:
directory).appendingPathComponent(file)

            //writing
            do {
                  try contents.write(to: path, atomically: false,
encoding: String.Encoding.utf8)
            }
            catch {print("Unable to write to: \(path)")}
      }

      print("Mode is greater than 50 numbers and your file is located
in Documents directory as: \(file)")
      return

    }
// end > array return

    if set2.count == 0 {
      print("Mode: \(set3)")
            return
      }
    else {
            print("Mode: \(set2)")
            return
      }
```

```
}
func collatz(_ set: [Double]) {
    var set2 = [Double](); set2.removeAll()
    var n: Double = 0
    var response: Double
    var start, end: Double

    print("Do you want a random number from your set or a custom one?
(R/C)")
    let response2 = stringinput().lowercased()
    if response2 == "r" {
     let j = UInt32(set.count - 1)
     let num = Int(arc4random_uniform(j))
     let i = set[num]
     print("Your random number is: \(i)")
     n = i
     start = CACurrentMediaTime()
     while n >= 1 {
          if n == 1 {
               set2.append(n)
               break
          }
          if n.truncatingRemainder(dividingBy: 2) == 1 {
               n = 3 * (n) + 1
               set2.append(n)
          }
          else {
               n = n / 2
               set2.append(2)
          }
          print(set2)
     }
     end = CACurrentMediaTime()
     print("Total time to collatz was: \(end - start)")

    }
    if response2 == "c" {
    print("Which number would you like to find 'one' from?")
    response = doubleinput()
     n = response
     start = CACurrentMediaTime()
     while n >= 1 {
          if n == 1 {
               set2.append(n)
               break
          }
          if n.truncatingRemainder(dividingBy: 2) == 1 {
               n = 3 * (n) + 1
               set2.append(n)
          }
```

```
            else {
                  n = n / 2
                  set2.append(2)
            }
            print(set2)
      }
      end = CACurrentMediaTime()
      print("Total time to collatz was: \(end - start)")
    }
}
//
//*****************************************************************************
// Function menus returns an integer corresponding to user selection
//
//*****************************************************************************
//Main Menu
func menu() -> Int
{
    var userchoice: Int
    print(" ")
    print("- ************************** -")
    print("- Operation           Option -")
    print("- Data Manipulation      1   -")
    print("- Quit                   0   -")
    print("- ************************** -")
    print(" ")
    print("Enter Option: ", terminator: "")
    userchoice = intinput()
    return userchoice
}
// Equation and Laws Menu
func menu3() -> Int
{
    var userchoice: Int
    print(" ")
    print("- ************************** -")
    print("- Operation           Option -")
    print("- Load New Set           1   -")
    print("- Preform Statistics     2   -")
    print("- Quit                   0   -")
    print("- ************************** -")
    print(" ")
    print("Enter Option: ", terminator: "")
    userchoice = intinput()
    return userchoice
}
//
//*****************************************************************************
// Various While loop realting to their specific Menu
//
```

```
//*************************************************************************
func DataManipulation() {
    var arraysize: Int = 0
    var upperlimit: UInt32 = 0
    var response: String
    var mum: Bool = false

    print("Welcome to the Statistics Function.")
    var set = [Double] (repeating: 0.0, count: 0)
    print("\r Preparing to Load Set...")

    print(" ")
    while !mum {
     print("Would you like to load a random or custom set? (R/C)")
     response = strinput().lowercased()

     if response == "r" {

          let array = RandomSet()
          set = array.set
          arraysize = array.arraysize
          upperlimit = array.upperlimit
          mum = true
     }

     if response == "c" {

          let array = CustomSet()
          set = array.set
          arraysize = array.arraysize
          upperlimit = array.upperlimit
          mum = true
     }
     if mum == false {
          print("That is not a valid option")
          print("")
     }
    }
while !done3 {
    let select = menu3()

    switch(select) {

    case 0: //Quit Case
     print(" ")
     print("Goodbye!")
     print(" ")
     done = true
     done3 = true
     break
```

```
    case 1: // Load Array
     print(" ")
     print("Would you like to load a random or custom set? (R/C)")
     let response = stringinput().lowercased()

     if response == "r" {
          set.removeAll()
          let array = RandomSet()
          set = array.set
          arraysize = array.arraysize
          upperlimit = array.upperlimit

     }
     else {
          set.removeAll()
          let array = CustomSet()
          set = array.set
          arraysize = array.arraysize
          upperlimit = array.upperlimit

     }
     break
    case 2: // Stats
     average(set)
     Median(set)
     Mode(set, arraysize: arraysize, Upperlimit: upperlimit)
     MinandMax(set)
     StandDev2_Variance(set)
     print("Would you like to do The Collatz conjecture on your set of
numbers (Y/N)?")
     let response = stringinput().lowercased()
     if response == "y" {
          collatz(set)
     }
     break
    case 3: //back case
     done3 = true
     break
    default: print("Pick a better number")
        }
    }
}
// Data Menu
//
**************************************************************************
while !done {
    let select = menu()

    switch(select) {
```

```
    case 0: //Quit Case
      print("Goodbye!")
      print(" ")
      done = true
      done3 = true
         break

    case 1: // Main Data Manipulation
      DataManipulation()
         done3 = false
         break

    default: print("Please pick a better number")
    }
} // Main Menu
//
*****************************************************************/
```