Ethan Roberts

Ruby Assignment Output

CS 417

Roberts.E.assn05

**Problem 1** (Source-code and output)

# Ethan Roberts

# CS 417 Topics in OOP

# This script will loop through an array and

# count how many times a "key" is found inside array

# Ruby Assignment

**def** **occurences(**list**,**key**)**

keyMatchCount **=** 0

list**.**each **do** **|**num**|**

**if** num **==** key

keyMatchCount **=** keyMatchCount **+** 1

**end**

**end**

**return** keyMatchCount

**end**

myAry **=** **[**10**,**20**,**30**,**0**,**50**,**20**,**20**,**70**,**80**,**80**,**72**,**90**,**100**,**90**,**20**,**100**,**100**,**100**,**50**,**50**]**

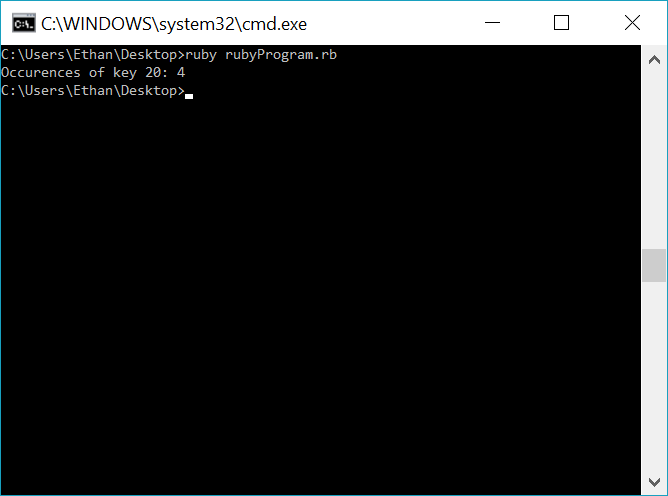
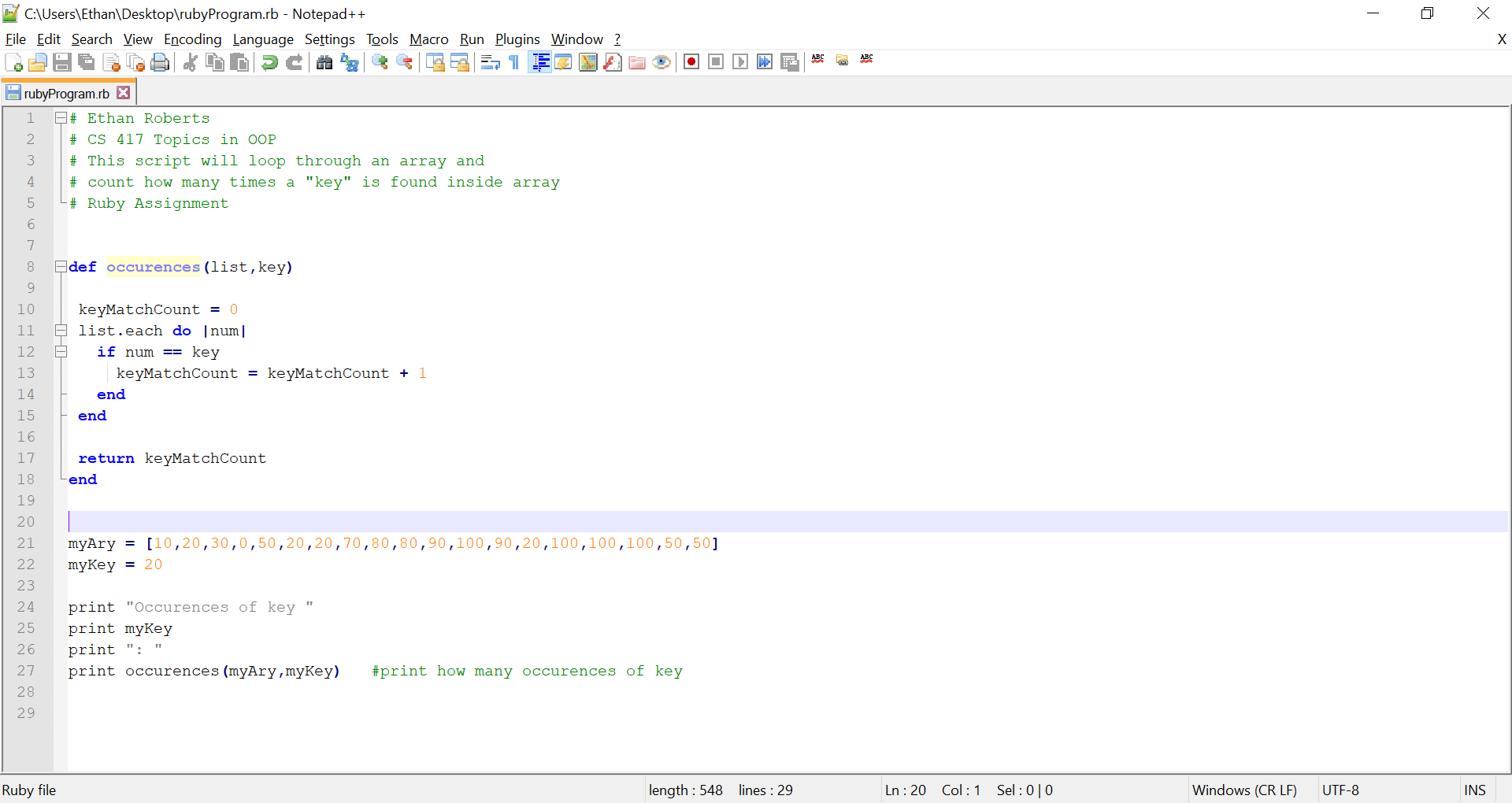
myKey **=** 94

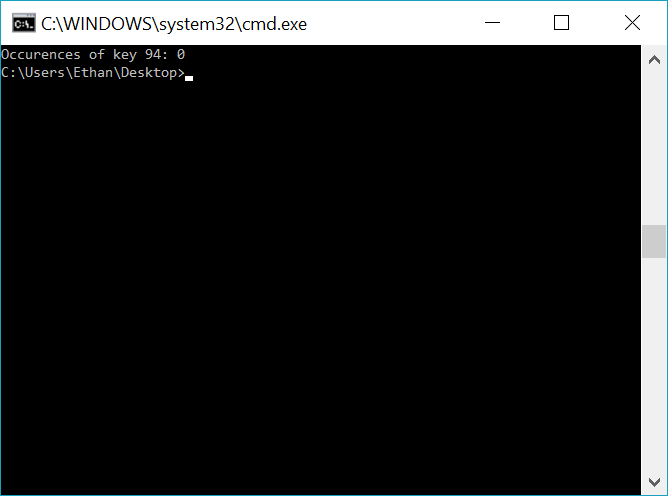
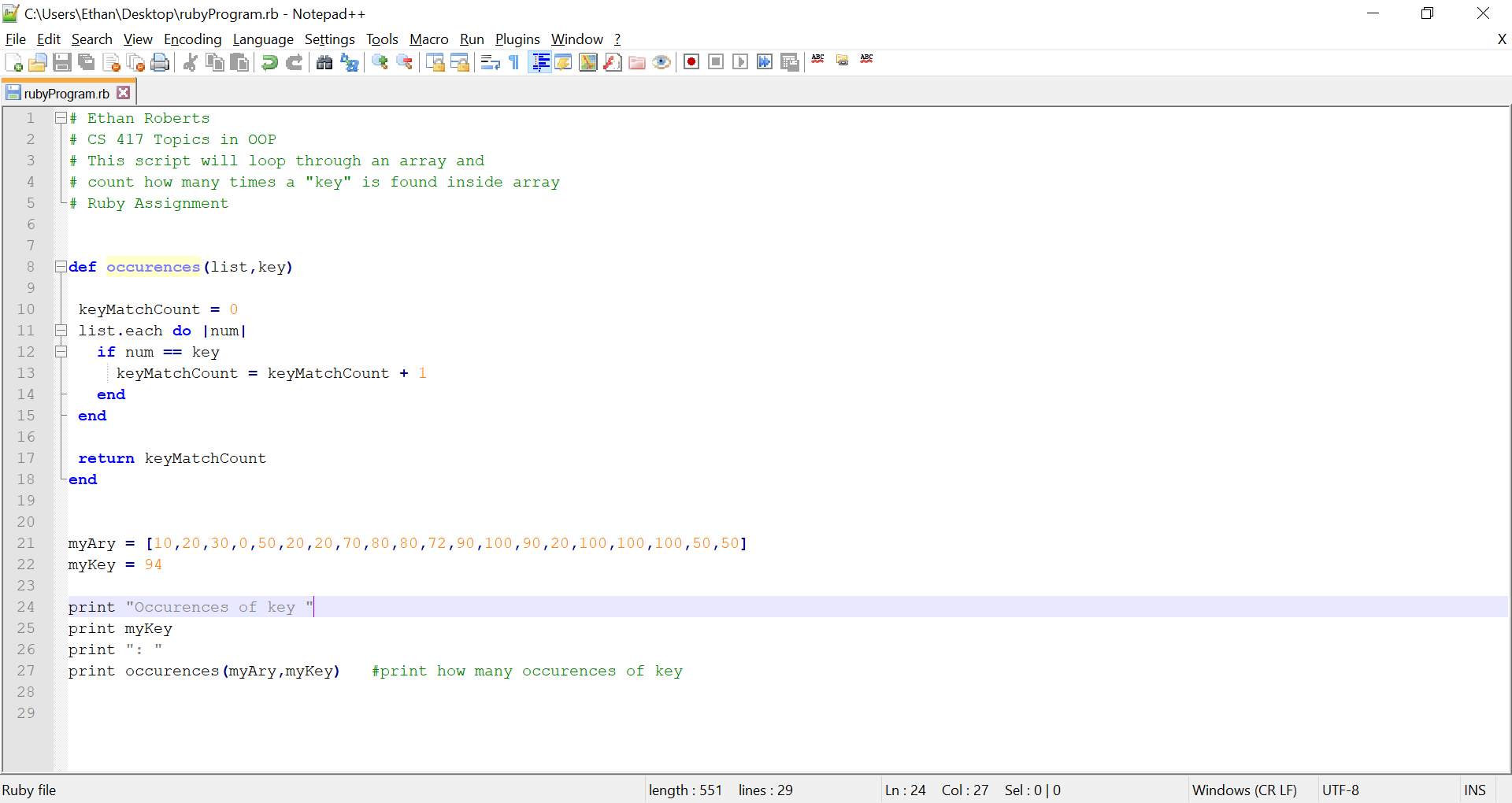
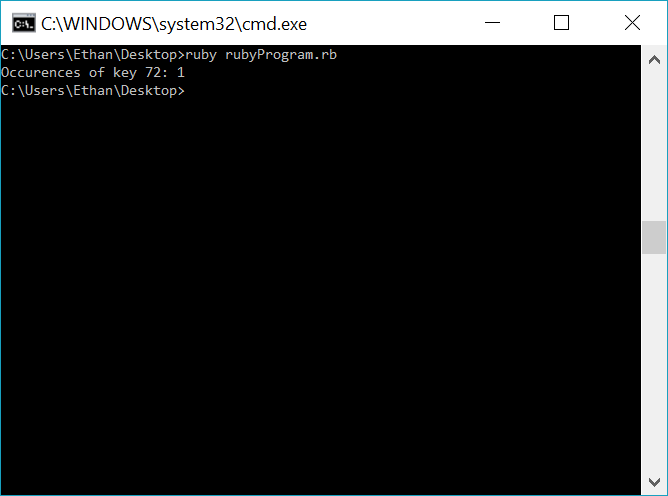
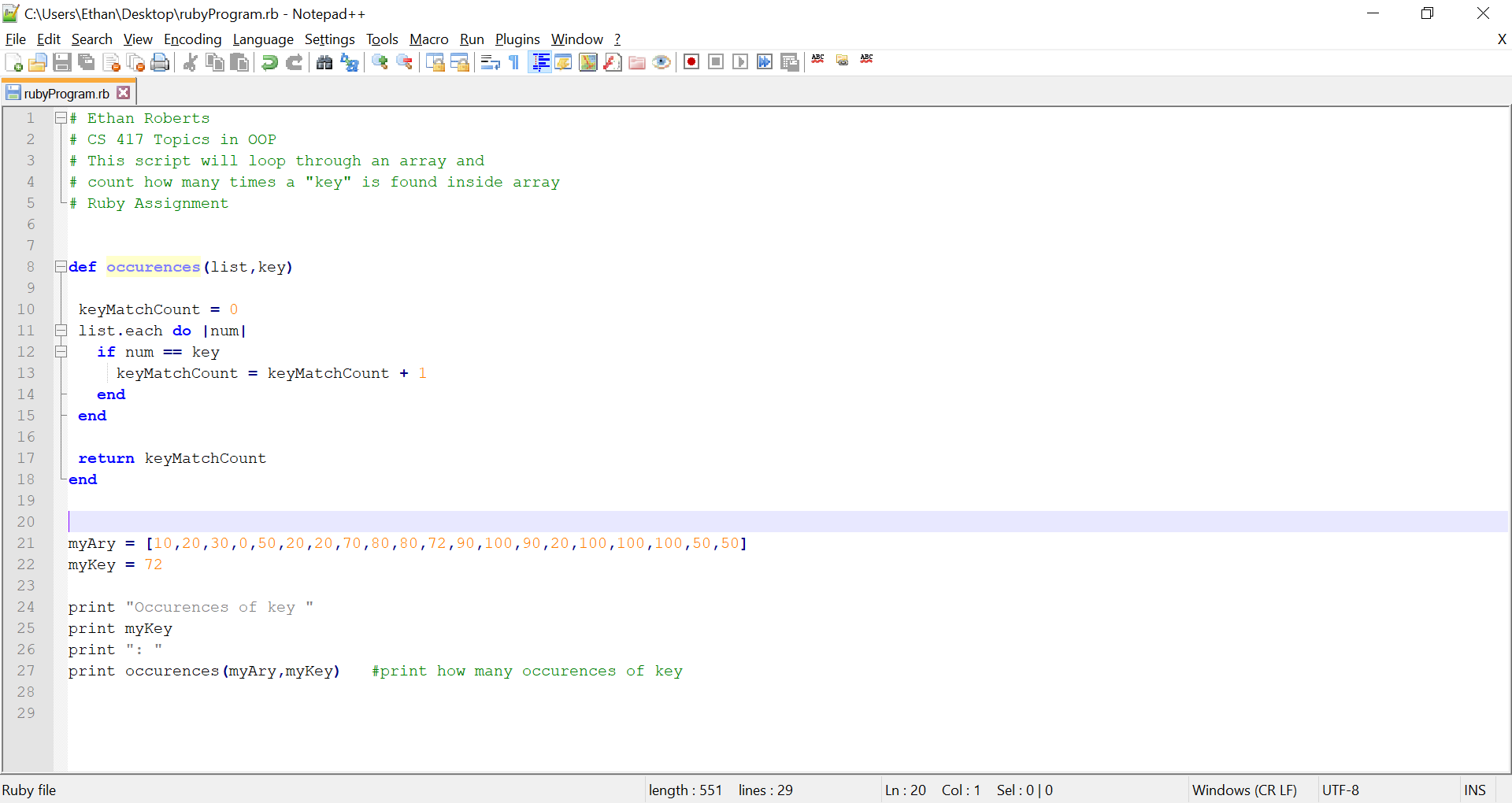
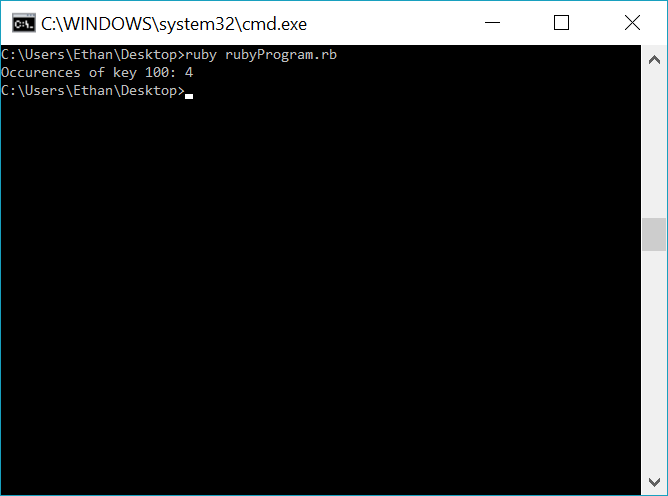
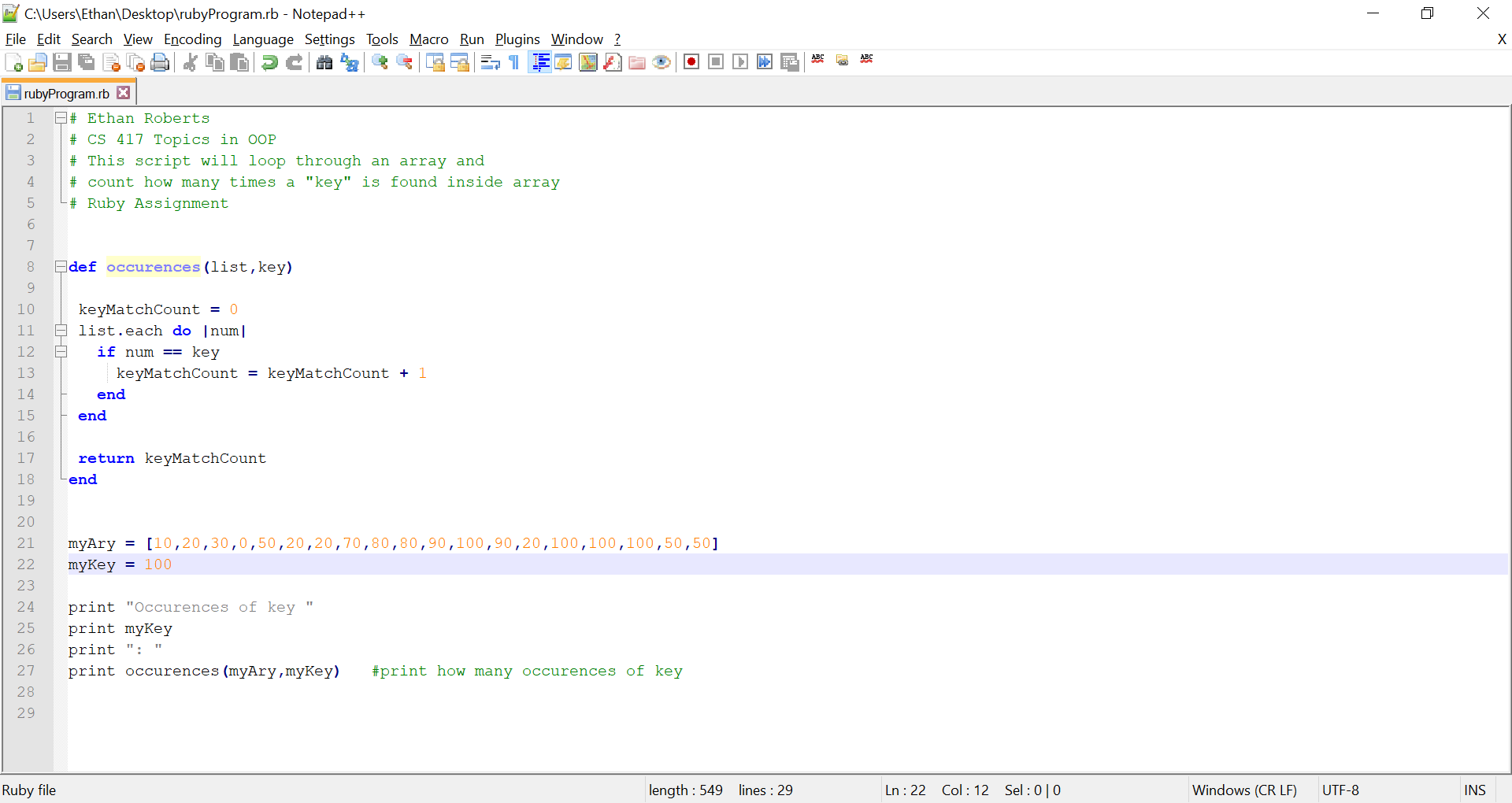
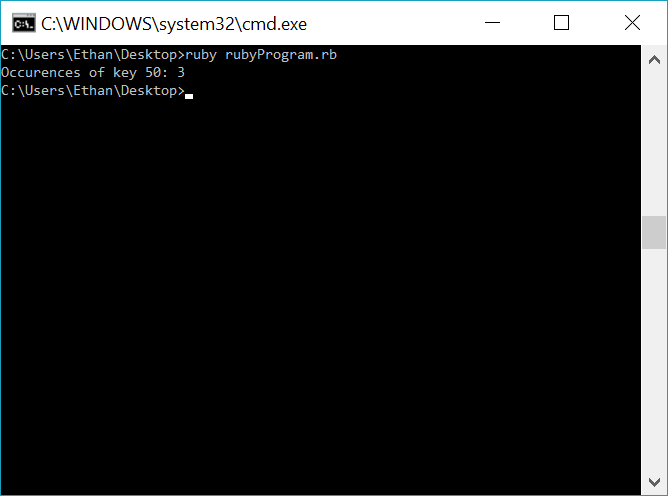
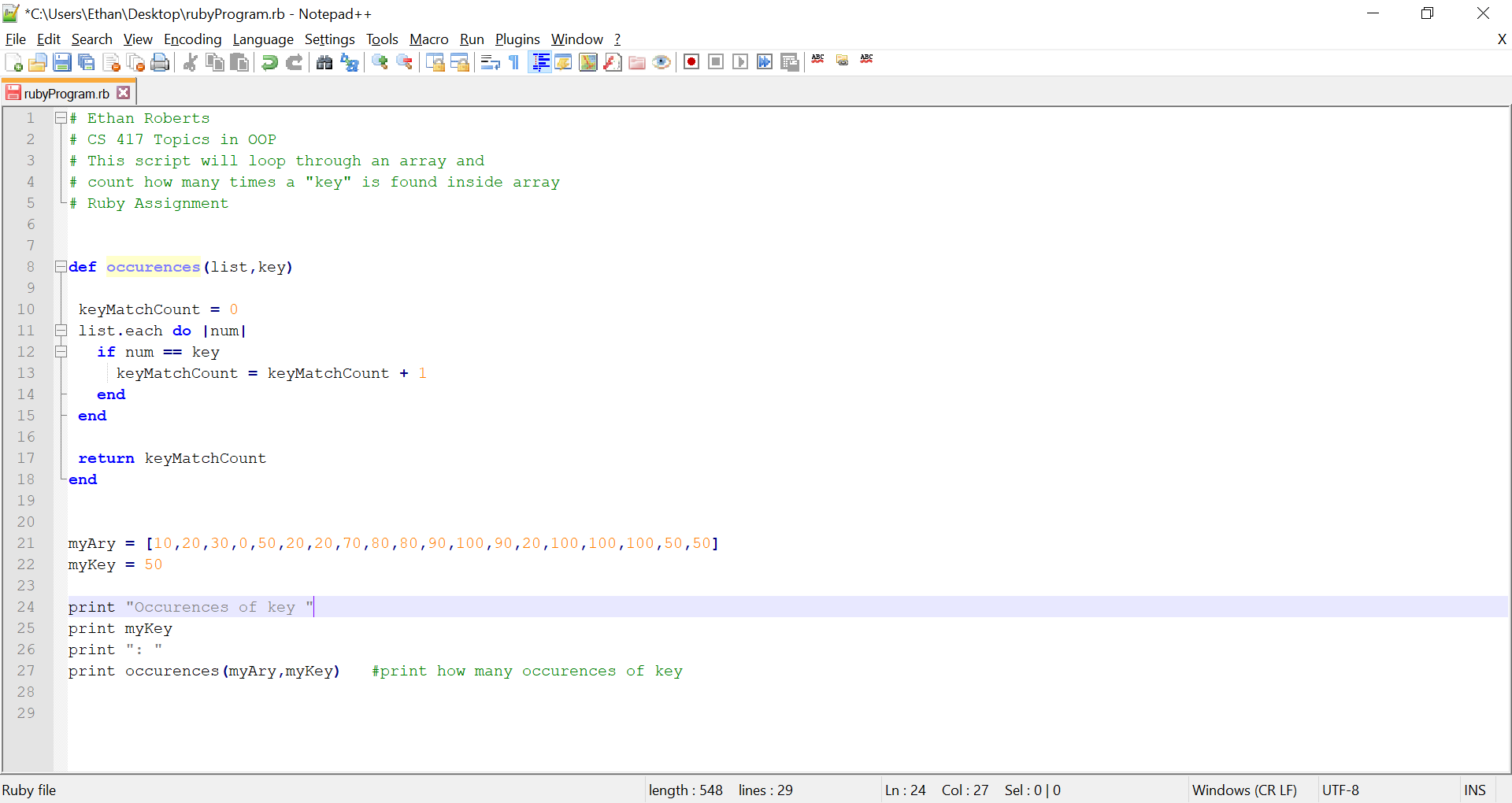
print "Occurences of key "

print myKey

print ": "

print occurences**(**myAry**,**myKey**)** #print how many occurences of key





**Problem 2** (Source-code and output)

# Ethan Roberts

# CS 417 Topics in OOP

# This script will loop through an array and return

# number of items that are less than item

# Ruby Assignment

**def** **numSmaller(**list**,**item**)**

keyMatchCount **=** 0

**if** list**.**length**()** **==** 0 #Making sure the array is not empty

print "Array is empty, program terminated."

**return**

**else**

list**.**each **do** **|**num**|**

**if** num **<** item

keyMatchCount **=** keyMatchCount **+** 1

**end**

**end**

**end**

**return** keyMatchCount

**end**

myAry **=** **[**0**,**1**,**2**,**3**,**4**,**5**,**6**,**7**,**8**,**9**,**10**]**

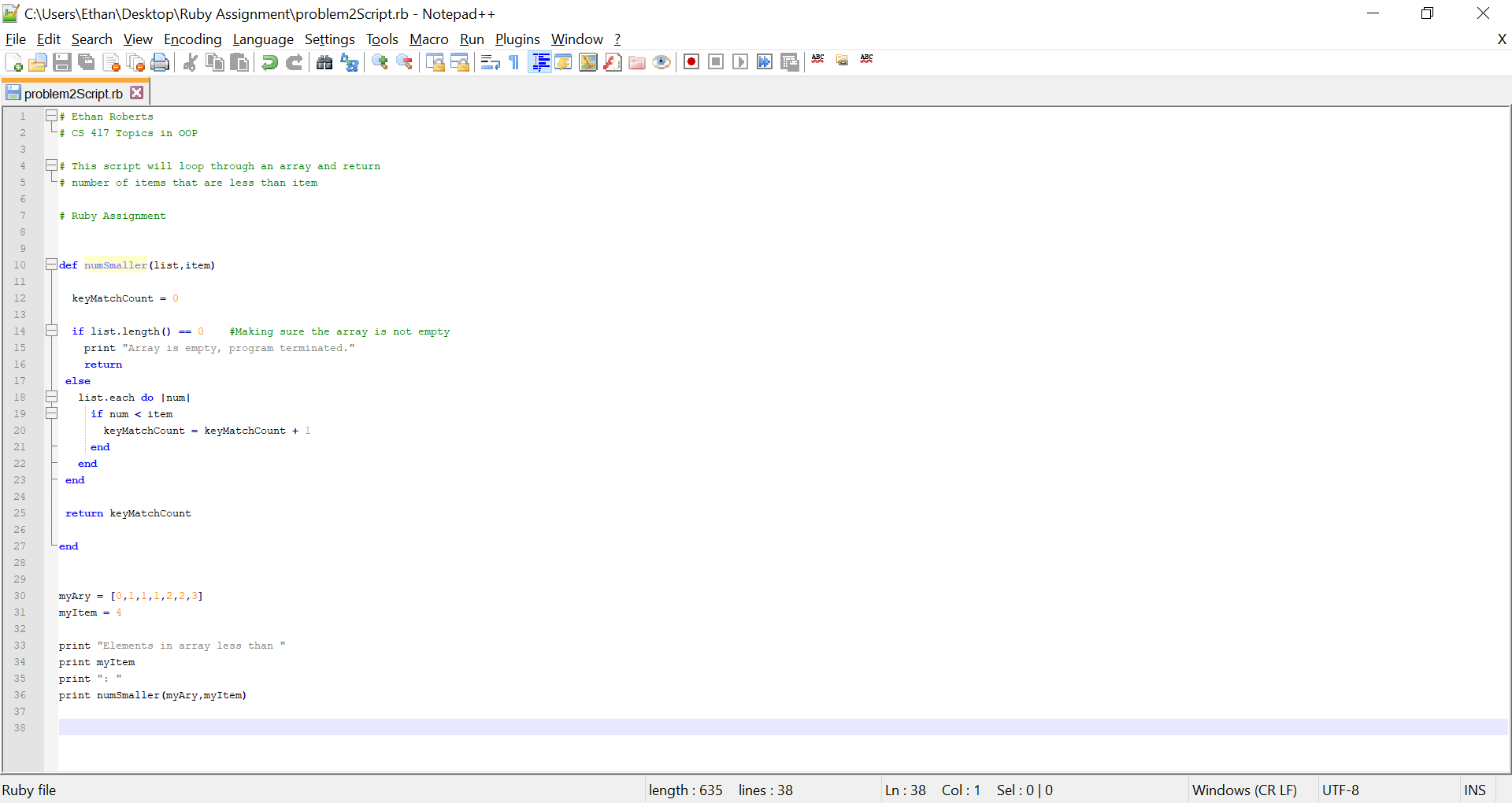
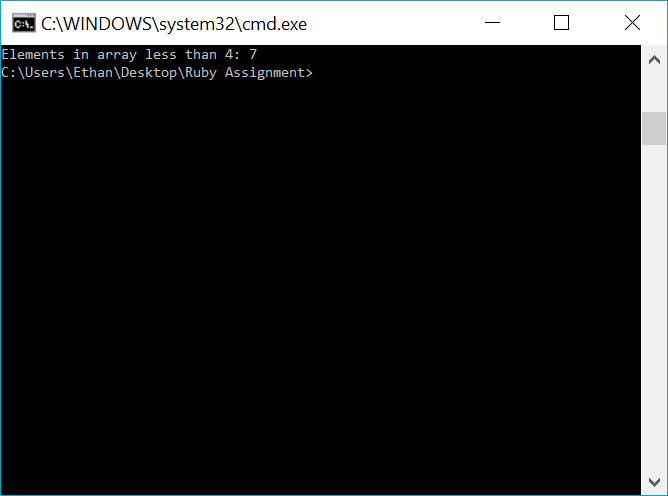
myItem **=** 8

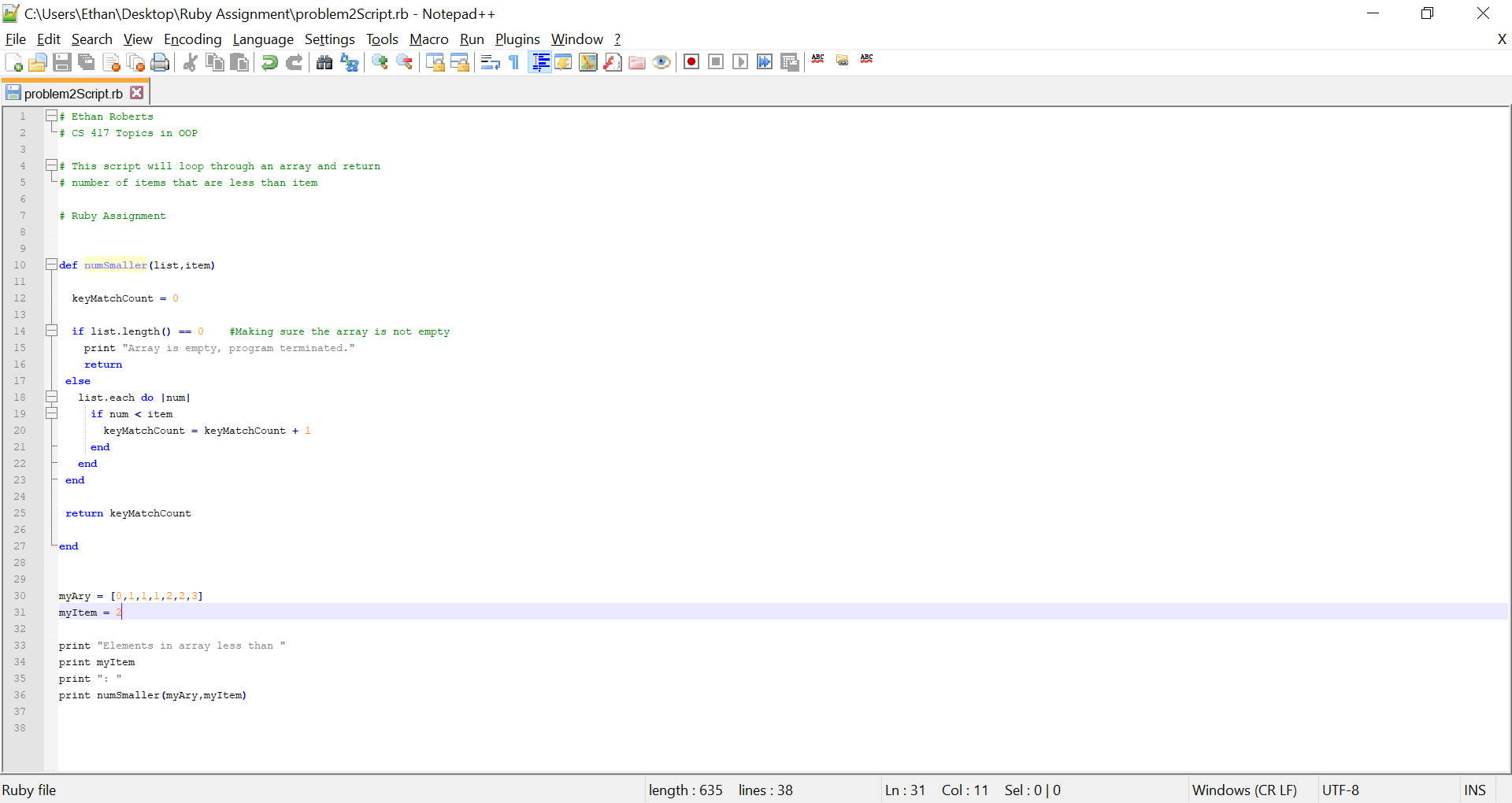
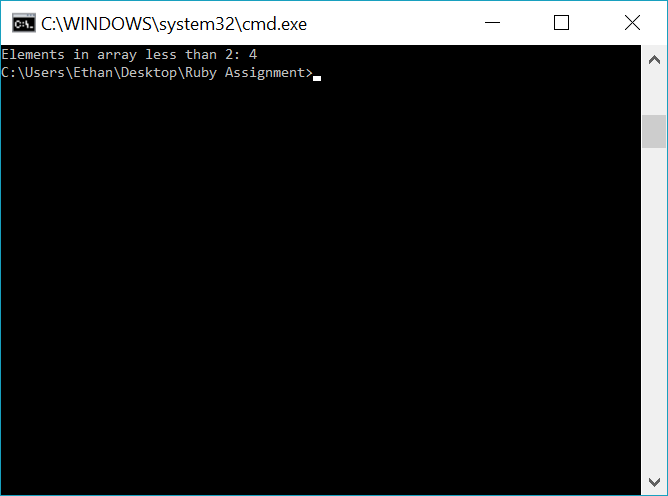
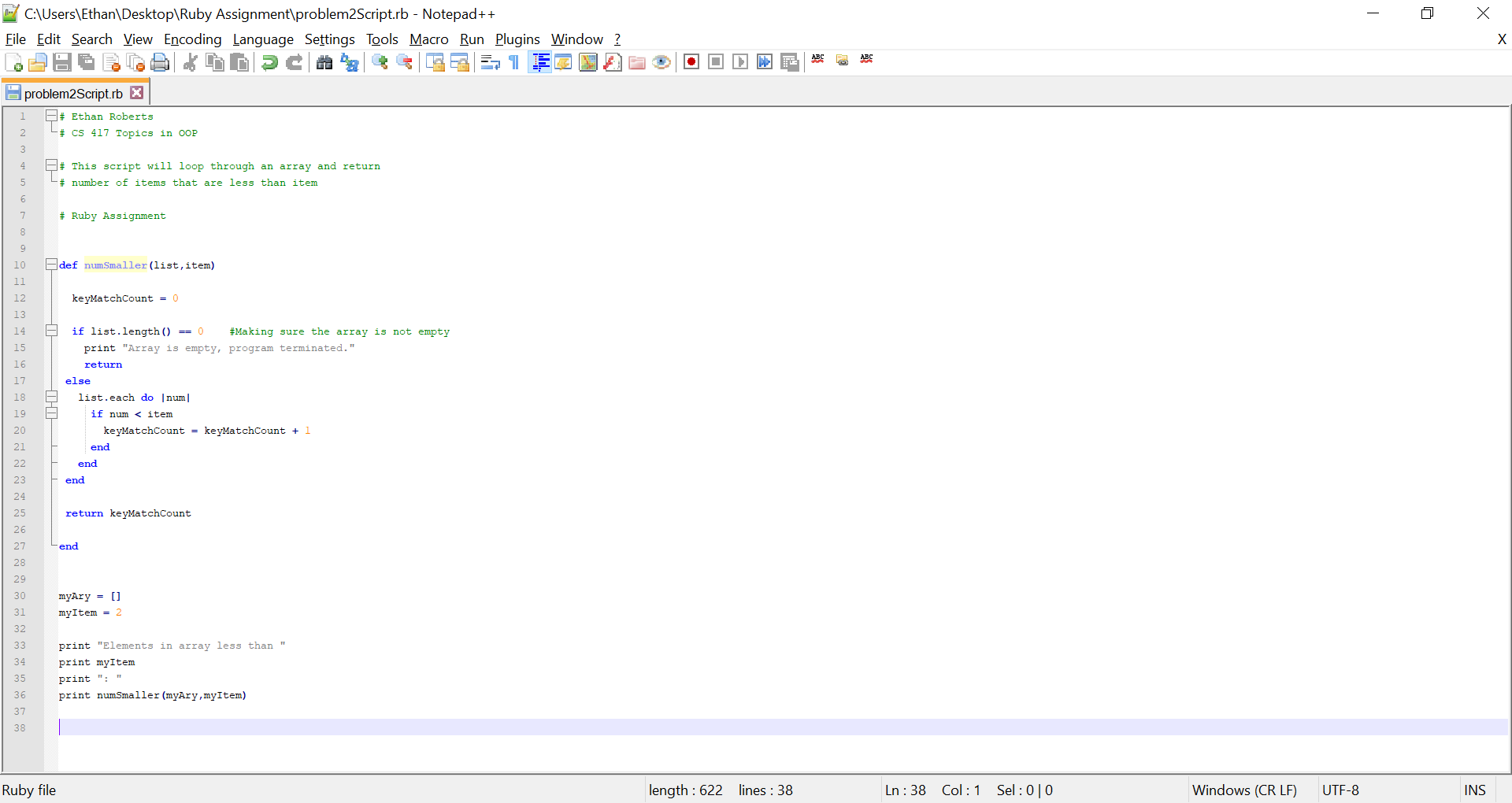
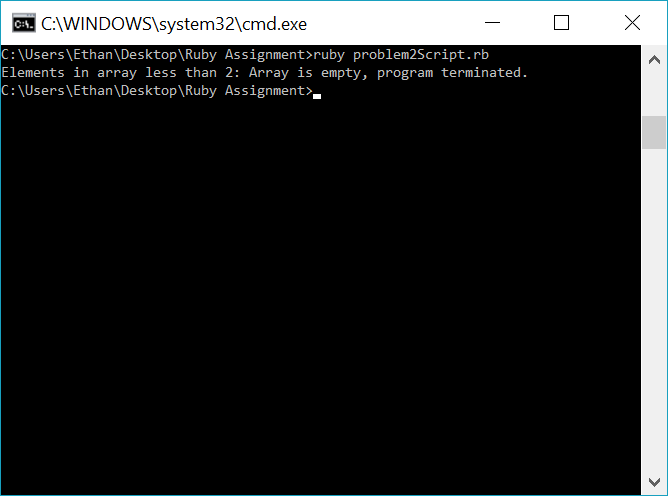
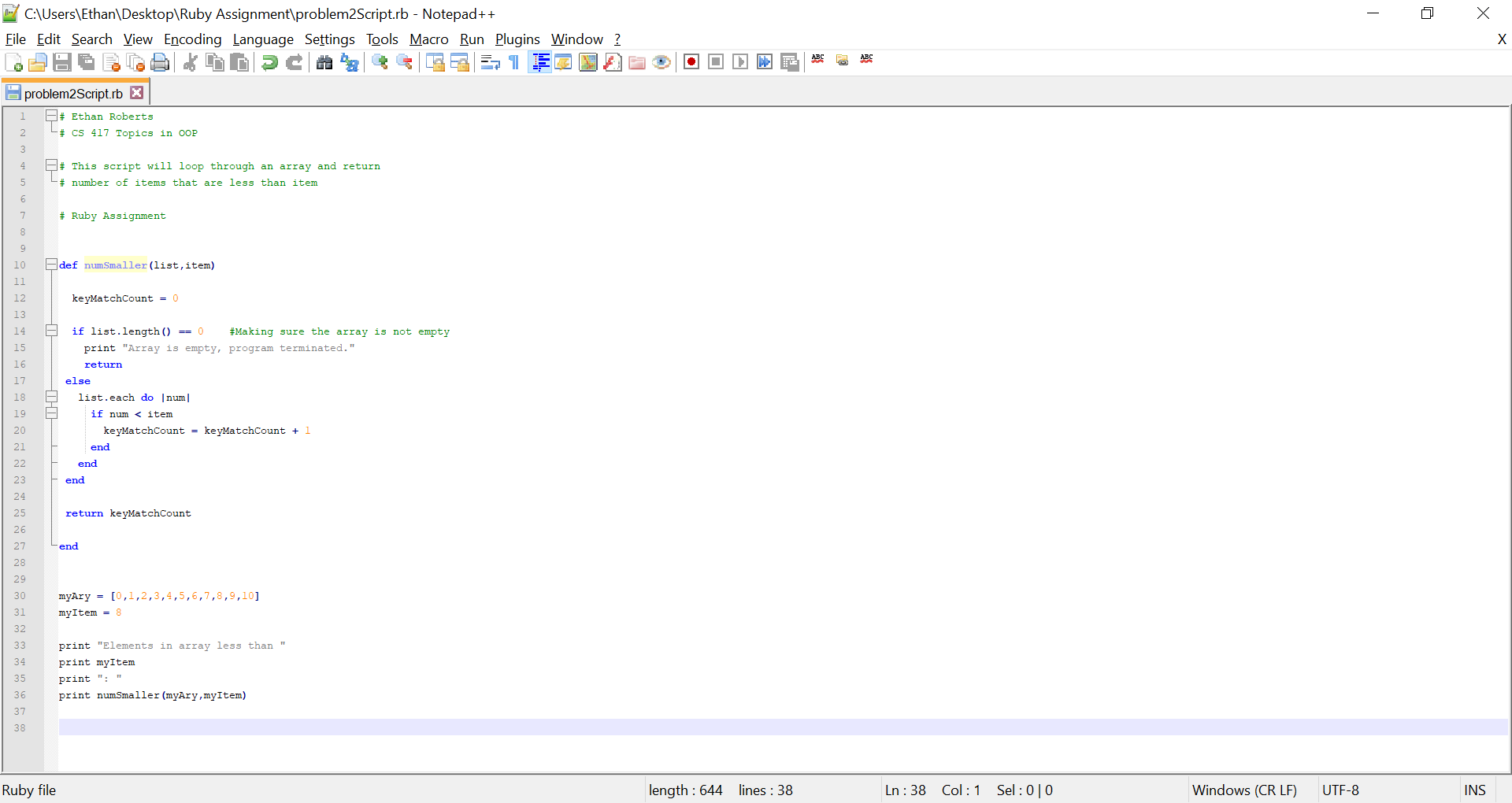
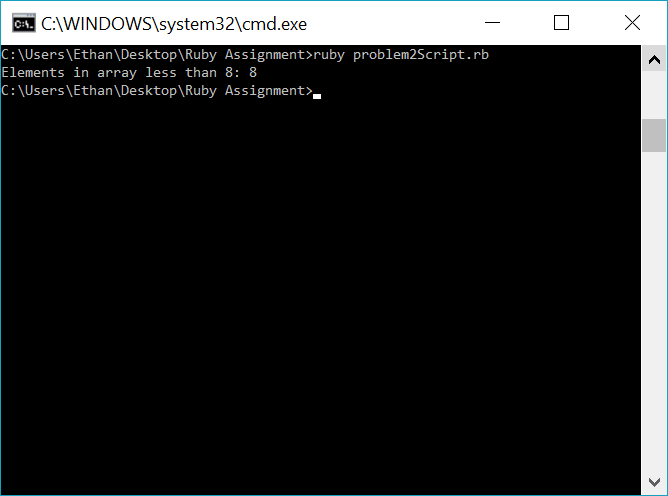
print "Elements in array less than "

print myItem

print ": "

print numSmaller**(**myAry**,**myItem**)**

**Problem 3** (Source-code and output)

# Ethan Roberts

# CS 417 Topics in OOP

# This script will find the second smallest index position.

# Ruby Assignment

# method for looping through array

# and seeing if smallest value has duplicate.

# If so, the second duplicate is second smallest

**def** **scanSmallestDuplicate(**smallestValue**,**list**)**

x **=** 0

dupCounter **=** 0

indexLocator **=** 0

**while** x **<** list**.**length **&&** dupCounter **<** 2

**if** list**[**x**]** **==** smallestValue

indexLocator **=** x

dupCounter **=** dupCounter **+** 1

**end**

x **=** x **+** 1

**end**

**if** dupCounter **>** 1

**return** indexLocator

**else**

**return** 0

**end**

**end**

**def** **secondSmallest(**list**)**

smallest **=** list**[**0**]**

secondSmallest **=** list**[**1**]**

temp **=** 0

secondSmallestIndex **=** 0

# Compare 0th element and 1st element

# in array

**if** secondSmallest **<** smallest

temp **=** smallest

smallest **=** secondSmallest

secondSmallest **=** temp

secondSmallestIndex **=** 1 #referencing index 1 of array

**end**

i **=** 2 # index 0 and 1 already compared

**while** i **<** list**.**length

**if** list**[**i**]** **<** secondSmallest

**if** list**[**i**]** **<** smallest

temp **=** smallest

smallest **=** list**[**i**]**

secondSmallest **=** temp

**else**

secondSmallest **=** list**[**i**]**

**end**

**end**

i **=** i **+**1

**end**

smallDupIndexLocator **=** scanSmallestDuplicate**(**smallest**,**list**)**

**if** smallDupIndexLocator **>** 0

**return** smallDupIndexLocator

**else**

**return** list**.**index**(**secondSmallest**)**

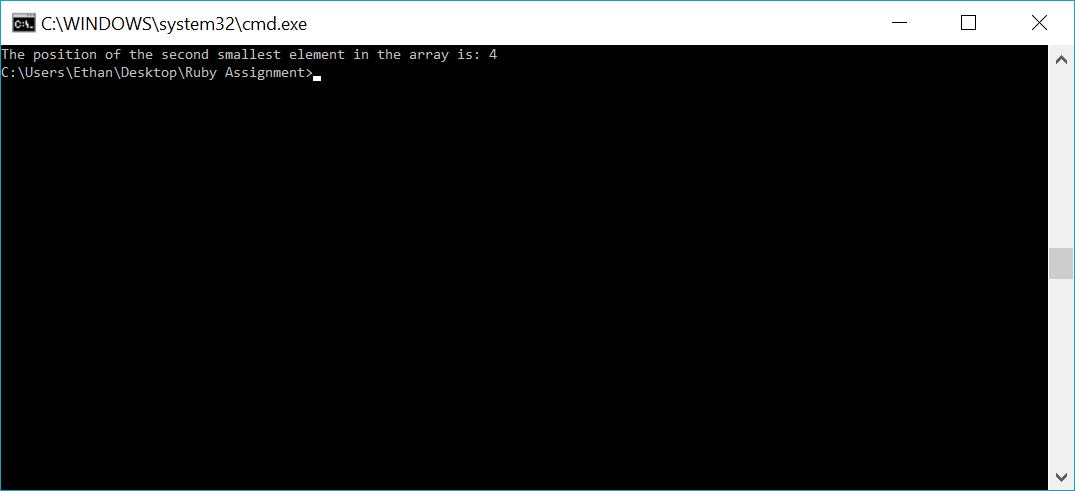
**end**

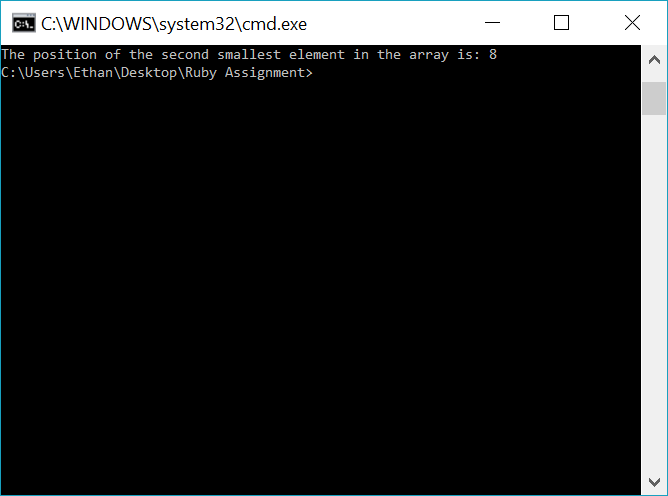
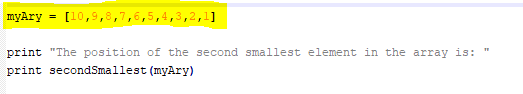
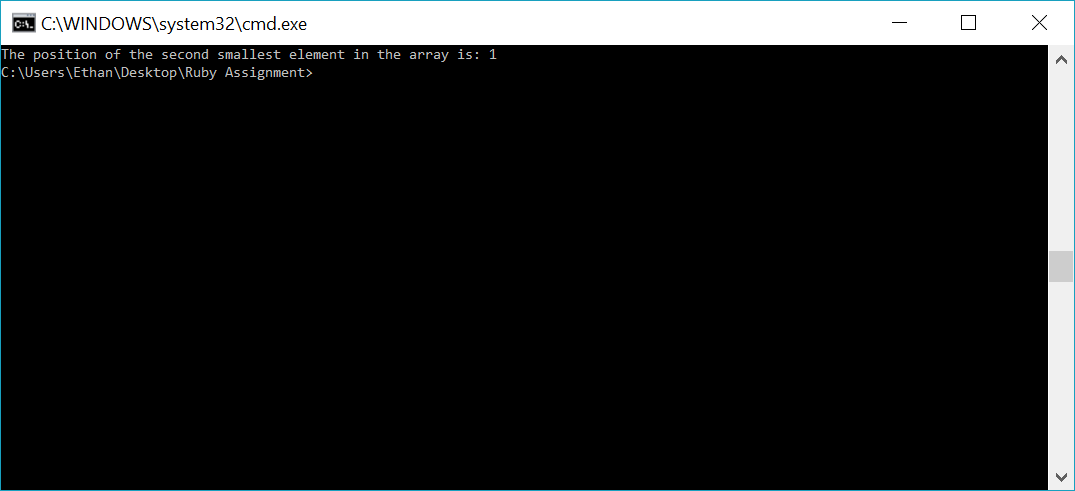
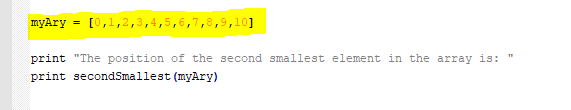
**end**

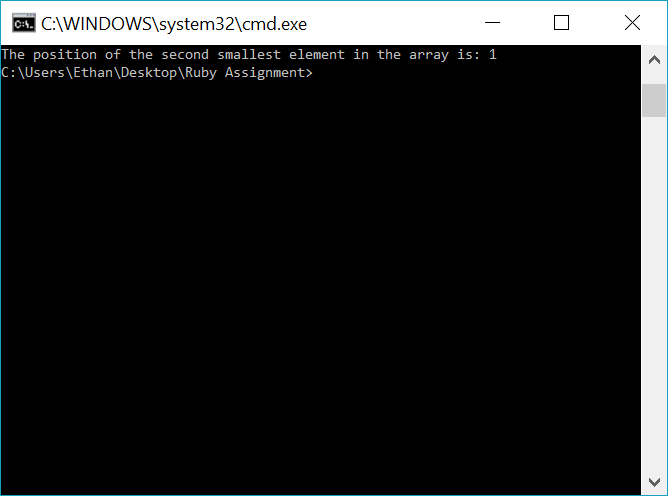
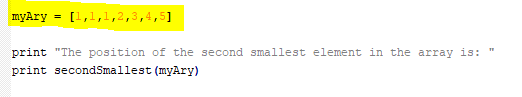
myAry **=** **[**9**,**8**,**6**,**22**,**4**,**6**,**77**,**4**]**

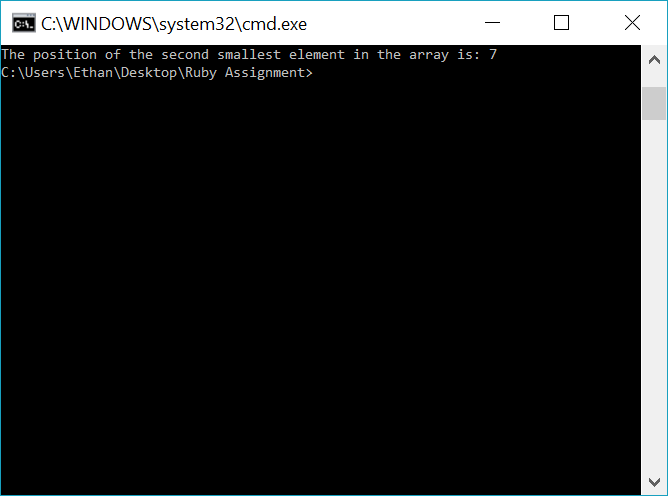
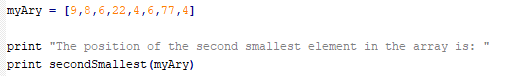
print "The position of the second smallest element in the array is: "

print secondSmallest**(**myAry**)**









**Problem 4** (Source-code and output)

# Ethan Roberts

# CS 417 Topics in OOP

# This script will contain two different

# methods for iterating over an array

# Ruby Assignment

**class** **Array**

#NOTE: Used source: "https://stackoverflow.com/questions/16422872/reimplementing-enumerable-map-method-in-ruby"

# for help understanding how passing a block works and for syntax help

**def** **iterativeMap**

out **=** **[]**

**if** block\_given?

**self.**each **{** **|**n**|** out **<<** **yield(**n**)}**

**return** out

**end**

**end**

**def** **recursiveMap**

**if** **self.**empty?

**return**

**else**

number **=** **self.**pop

number **=** number **\*** 2

recursiveMap

**end**

**return** **self** **<<** number #returning array with calculated values

**end**

**end**

answerAry **=** **[]** # for holding final answer

#Section for testing "iterativeMap" method

myAry **=** **[**4**,**3**,**2**,**1**]**

answerAry **=** myAry**.**iterativeMap**{|**n**|** n **\*** 2**}**

print "\nUsing iterative method: "

print answerAry

print "\n\n"

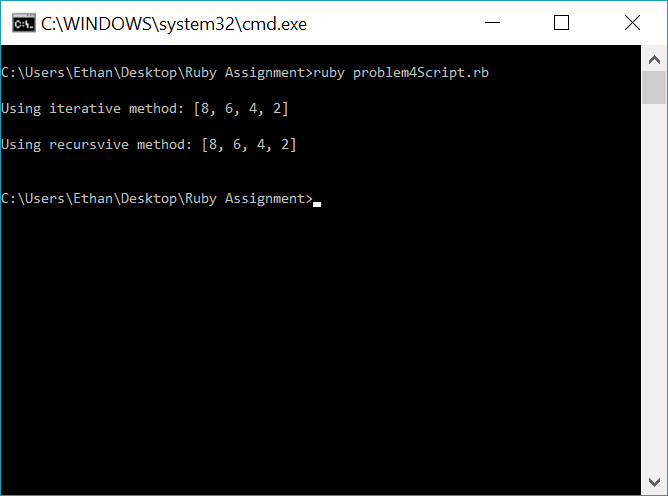
#Section for testing "recursiveMap" method

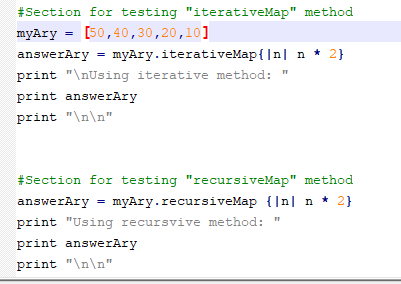
answerAry **=** myAry**.**recursiveMap **{|**n**|** n **\*** 2**}**

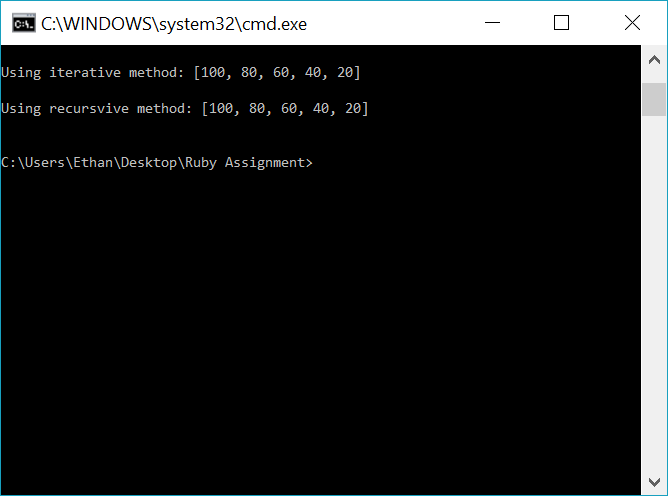
print "Using recursvive method: "

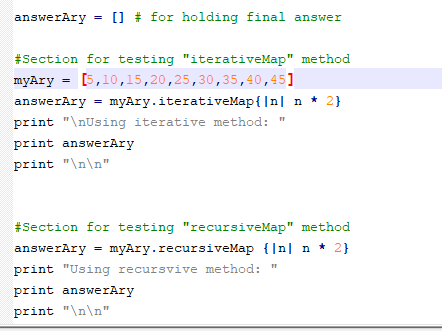
print answerAry

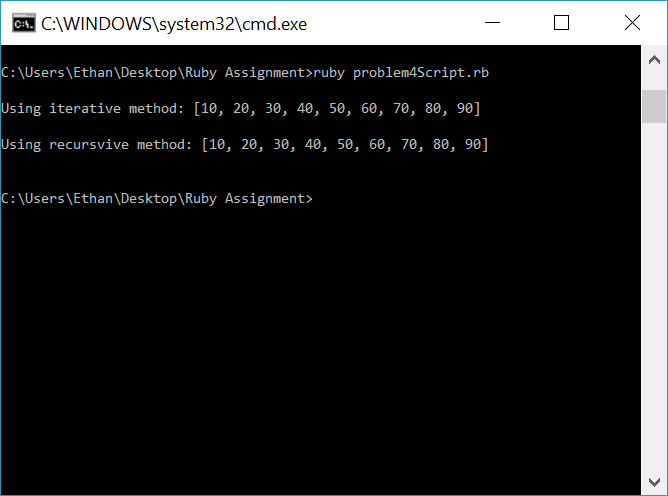
print "\n\n"











**Problem 5** (Source-code and output)

# Ethan Roberts

# CS 417 Topics in OOP

# This script will manipulates a list

# Ruby Assignment

# To understand instance variables for classes,

# I used: http://ruby-for-beginners.rubymonstas.org/writing\_classes/instance\_variables.html

**class** **List**

**def** **initialize** # self-note: initialize is a ruby-defined "constructor"

@backingStore **=** **[]** #@ sign means instance variable

**end**

**def** **insert(**value**)**

@backingStore**.**push**(**value**)**

**end**

# cite: https://ruby-doc.org/core-2.2.0/Array.html#method-i-index

# implemented ".index" method from this site

**def** **delete(**value**)**

**if** @backingStore**.**length**()** **==** 0 # backingStore is empty

**return**

**else**

@backingStore**.**each **do** **|**n**|**

**if** n **==** value

@backingStore**[**@backingStore**.**index**(**n**)]** **=** **nil**

**end**

**end**

**end**

**end**

**def** **traverse**

@backingStore**.**each **do** **|**n**|**

print n

print " "

**end**

**end**

**end**

#-------- BEGIN TESTING CODE ---------

myList **=** List**.**new

stringList **=** List**.**new

myList**.**insert**(**1**)**

myList**.**insert**(**2**)**

myList**.**insert**(**3**)**

myList**.**insert**(**4**)**

myList**.**insert**(**5**)**

myList**.**insert**(**6**)**

myList**.**insert**(**7**)**

myList**.**insert**(**8**)**

print "Printing full list before deletes: "

myList**.**traverse

print "\n"

myList**.**delete**(**4**)**

print "After deletion: "

myList**.**traverse

print "\n"

myList**.**delete**(**3**)**

myList**.**delete**(**7**)**

print "After deletion: "

myList**.**traverse

print "\n"

myList**.**delete**(**1**)**

print "After deletion: "

myList**.**traverse

print "\n"

myList**.**insert**(**988**)**

print "After insert: "

myList**.**traverse

print "\n\n\*\*Begin testing different data type...\n\n"

#---------- TESTING DIFFERENT OBJECT DATA-TYPE -------------

stringList**.**insert**(**"cat"**)**

stringList**.**insert**(**"dog"**)**

stringList**.**insert**(**"shark"**)**

stringList**.**insert**(**"parrot"**)**

stringList**.**insert**(**"lizard"**)**

stringList**.**insert**(**"tiger"**)**

stringList**.**insert**(**"pelican"**)**

print "Printing full list before deletes: "

stringList**.**traverse

print "\n"

stringList**.**delete**(**"pelican"**)**

print "After deletion: "

stringList**.**traverse

print "\n"

stringList**.**delete**(**"shark"**)**

print "After deletion: "

stringList**.**traverse

print "\n"

