1INF06 Data Structures and Methodical Programming

Loaiza Vasquez, Manuel Alejandro
Oyarce Tocto, Elizabeth Patricia
Saras Rivera, André Edgardo

Octubre 8, 2020

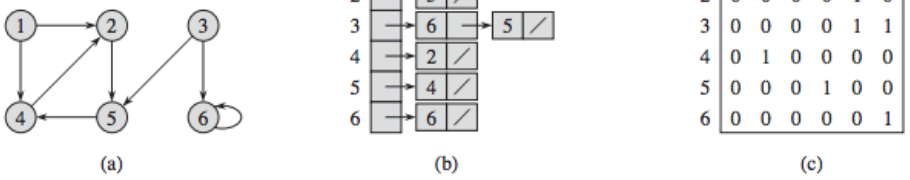
Pontificia Universidad Católica del Perú
Lima, Perú
manuel.loaiza@pucp.edu.pe
a20182778@pucp.edu.pe
andre.saras@pucp.edu.pe

Third report of the course Data Structures and Methodical Programming taught at the Faculty of Science and Engineering at Pontificia Universidad Católica del Perú (PUCP) by Viktor Khlebnikov.

Contents

1 Linked data structure research: Double Linked List

The structures we have to research and implement the following reports are graphs. There are two common ways to represent a graph: using and adjacency matrix or using an adjacency list. Given a graph $G = (V, E)$, the adjacency matrix representation is easier because we are going to create an array of $|V|^2$ units of memory and a_{ij} will represent if an edge between the nodes i and j exists. However, we need a lot of memory to represent the graph and, also, there will be too much unused space. As a consequence, the adjacency list is the most efficient, in terms of space complexity, way to represent a graph because we are going to create nodes on each list that represent each existing edge if necessary.

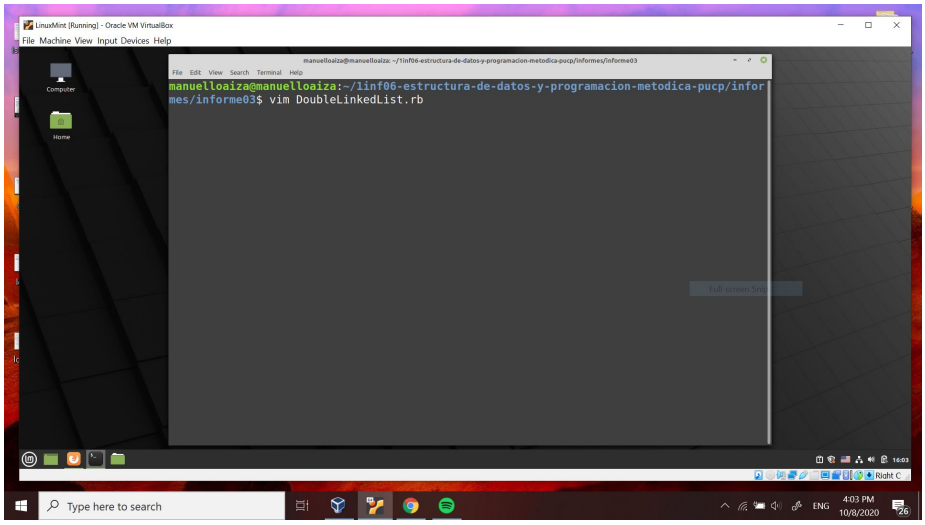


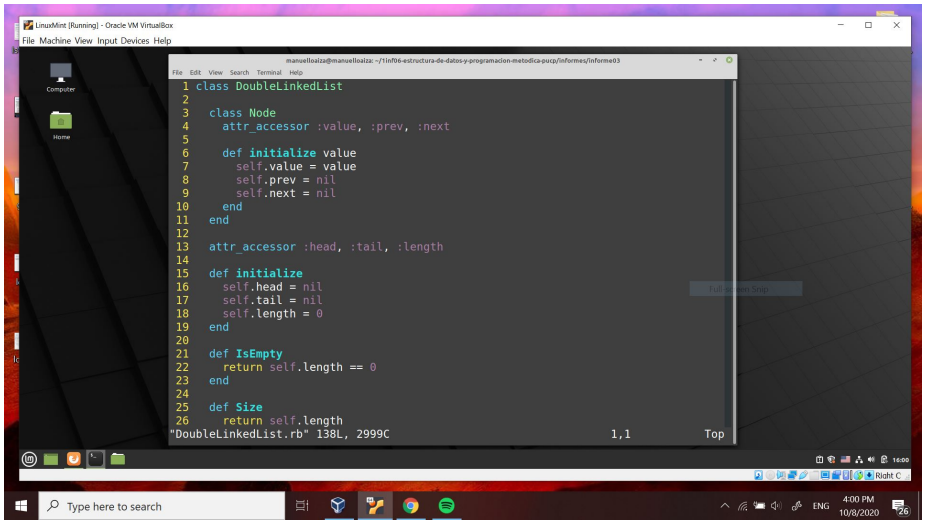
	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

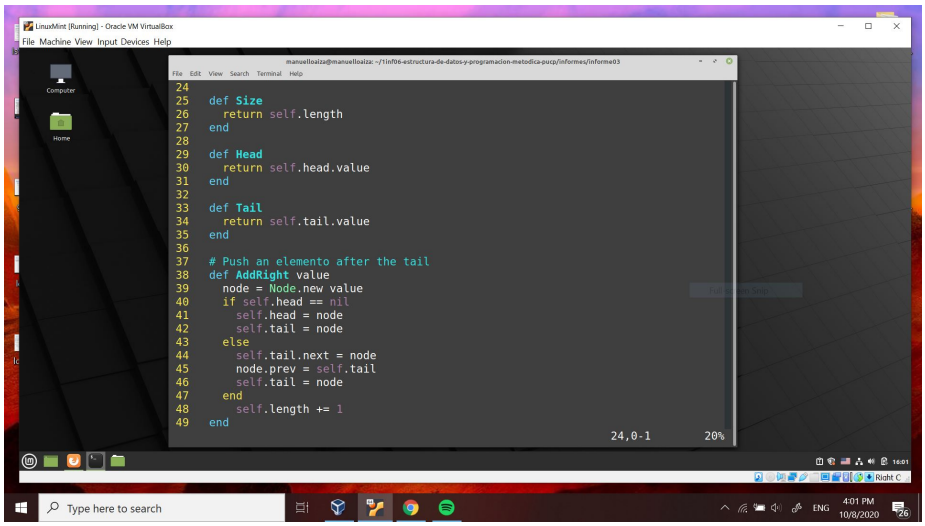
As we can see, we are going to represent a graph like the one in (a) using an array of double linked lists as shown in (b). First, we have to implement a double linked list in Ruby and then, each member is going to solve a Google coding interview problem using our data structure in our programs. In our next report, we are going to use this data structure to represent a graph and traverse it with algorithms like Depth First Search and Breadth First Search.

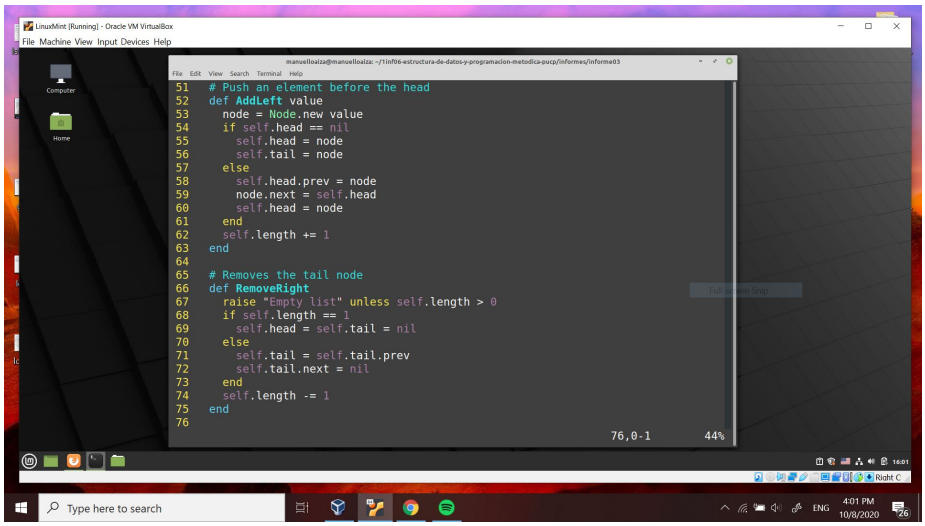
In the following pages, we are going to show the implementation of the double linked list data structure and the explanation is inside the code as comments. In `main.rb`, we have prepared some tests with assertions, which are called with the statement `raise` in Ruby, to check if our data structure has bugs. Also, the method `Print` that we have define is going to help to see how our list is growing or shrinking after each operation.

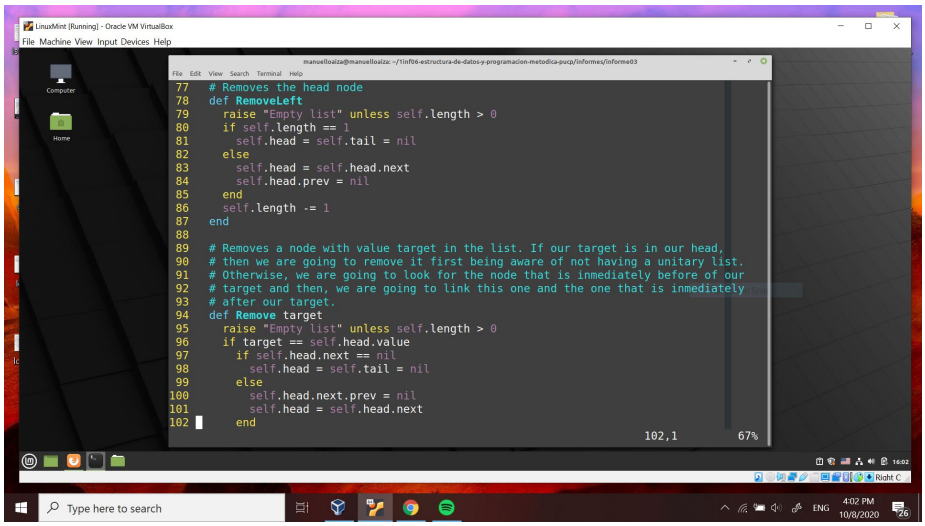
1.1 Implementation

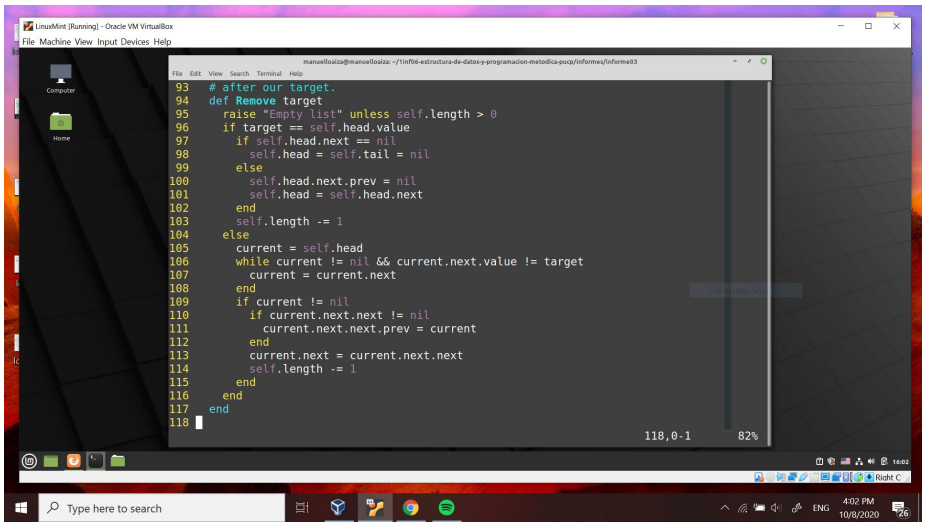


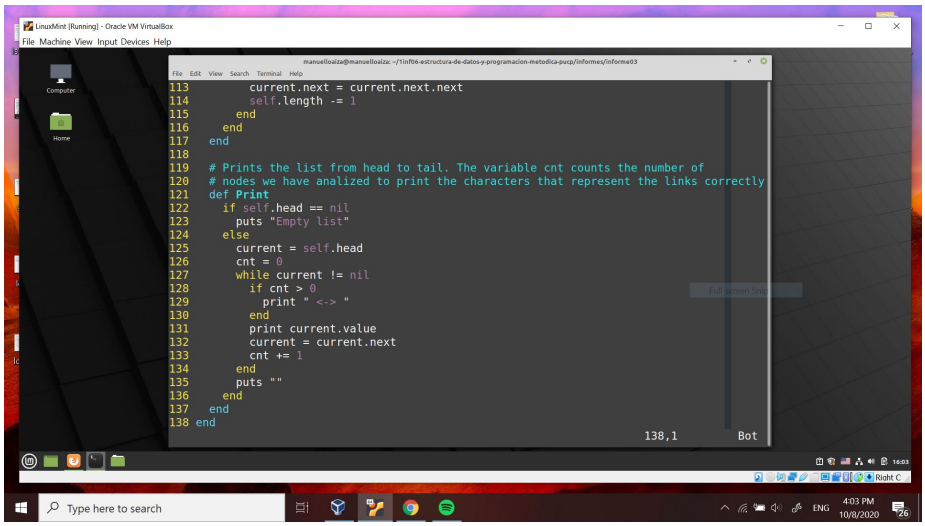



LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
manuelloaiza@manuelloaiza: ~/1inf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03
File Edit View Search Terminal Help
24
25 def Size
26 return self.length
27 end
28
29 def Head
30 return self.head.value
31 end
32
33 def Tail
34 return self.tail.value
35 end
36
37 # Push an elemento after the tail
38 def AddRight value
39 node = Node.new value
40 if self.head == nil
41 self.head = node
42 self.tail = node
43 else
44 self.tail.next = node
45 node.prev = self.tail
46 self.tail = node
47 end
48 self.length += 1
49 end
24,0-1 20%
Computer
Home
Type here to search
ENG
4:01 PM
10/8/2020

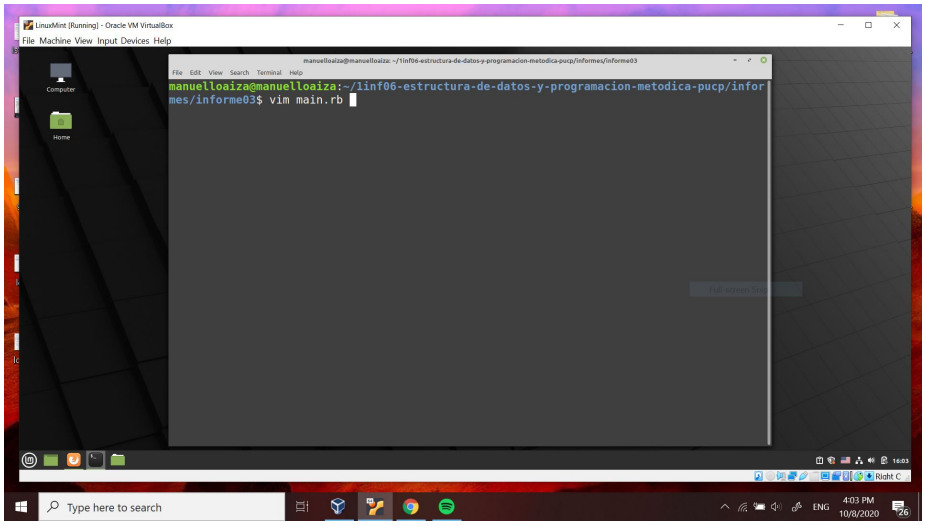

LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
manuelloaiza@manuelloaiza: ~/1inf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03
File Edit View Search Terminal Help
51 # Push an element before the head
52 def AddLeft value
53 node = Node.new value
54 if self.head == nil
55 self.head = node
56 self.tail = node
57 else
58 self.head.prev = node
59 node.next = self.head
60 self.head = node
61 end
62 self.length += 1
63 end
64
65 # Removes the tail node
66 def RemoveRight
67 raise "Empty list" unless self.length > 0
68 if self.length == 1
69 self.head = self.tail = nil
70 else
71 self.tail = self.tail.prev
72 self.tail.next = nil
73 end
74 self.length -= 1
75 end
76
76,0-1 44%
Computer
Home
Type here to search
ENG
4:01 PM
10/8/2020

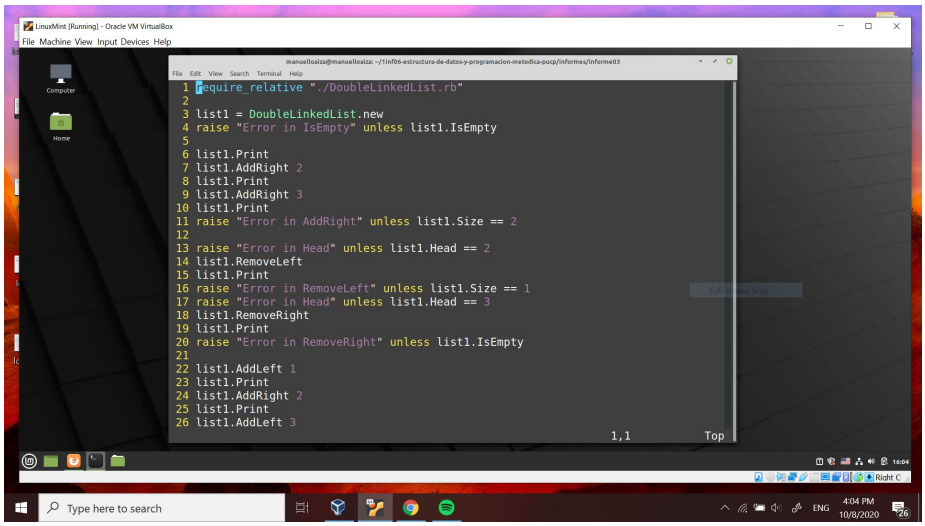

LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
File Edit View Search Terminal Help
77 # Removes the head node
78 def RemoveLeft
79 raise "Empty list" unless self.length > 0
80 if self.length == 1
81 self.head = self.tail = nil
82 else
83 self.head = self.head.next
84 self.head.prev = nil
85 end
86 self.length -= 1
87 end
88
89 # Removes a node with value target in the list. If our target is in our head,
90 # then we are going to remove it first being aware of not having a unitary list.
91 # Otherwise, we are going to look for the node that is inmediately before of our
92 # target and then, we are going to link this one and the one that is inmediately
93 # after our target.
94 def Remove target
95 raise "Empty list" unless self.length > 0
96 if target == self.head.value
97 if self.head.next == nil
98 self.head = self.tail = nil
99 else
100 self.head.next.prev = nil
101 self.head = self.head.next
102 end
102,1 67%
Computer
Home
Type here to search
ENG
4:02 PM
10/8/2020

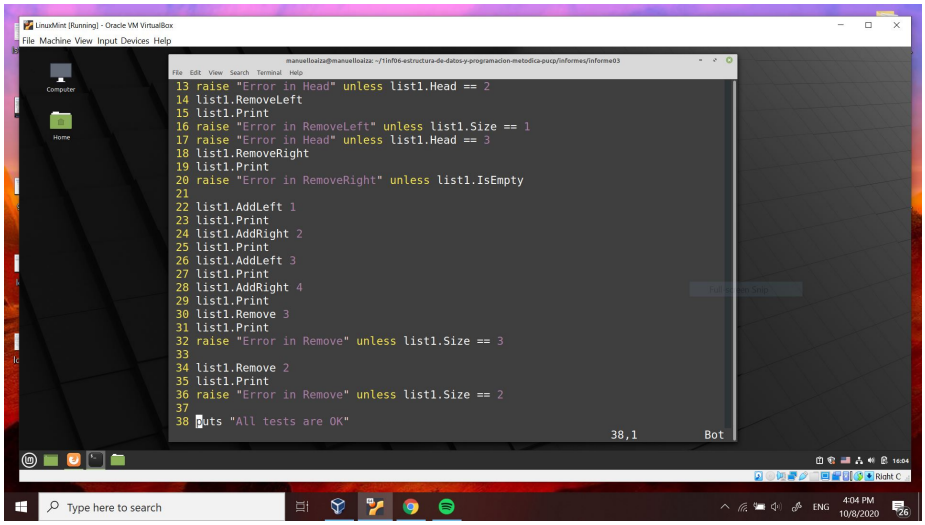

LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
File Edit View Search Terminal Help
93 # after our target.
94 def Remove target
95 raise "Empty list" unless self.length > 0
96 if target == self.head.value
97 if self.head.next == nil
98 self.head = self.tail = nil
99 else
100 self.head.next.prev = nil
101 self.head = self.head.next
102 end
103 self.length -= 1
104 else
105 current = self.head
106 while current != nil && current.next.value != target
107 current = current.next
108 end
109 if current != nil
110 if current.next.next != nil
111 current.next.next.prev = current
112 end
113 current.next = current.next.next
114 self.length -= 1
115 end
116 end
117 end
118
118,0-1 82%
Computer
Home
Type here to search
ENG
4:02 PM
10/8/2020

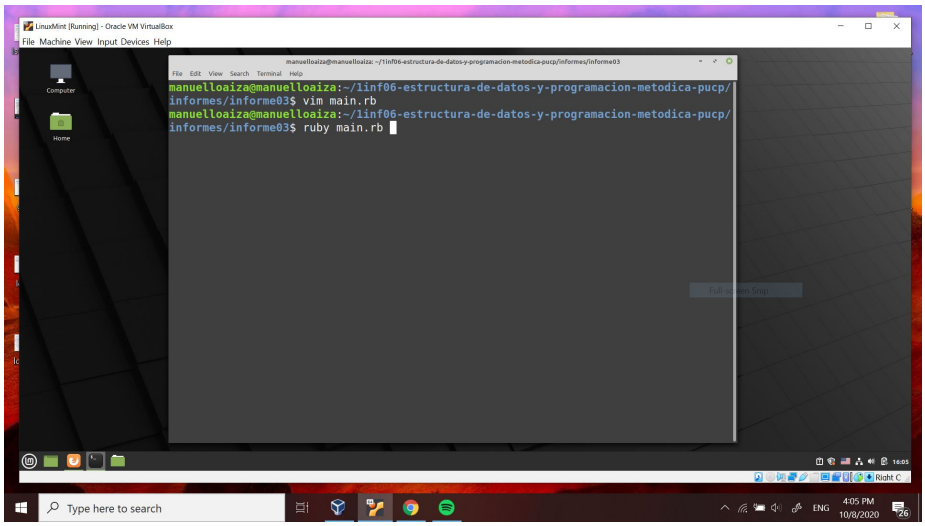

LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Computer
Home
File Edit View Search Terminal Help
113 current.next = current.next.next
114 self.length -= 1
115 end
116 end
117 end
118
119 # Prints the list from head to tail. The variable cnt counts the number of
120 # nodes we have analized to print the characters that represent the links correctly
121 def Print
122 if self.head == nil
123 puts "Empty list"
124 else
125 current = self.head
126 cnt = 0
127 while current != nil
128 if cnt > 0
129 print " <-> "
130 end
131 print current.value
132 current = current.next
133 cnt += 1
134 end
135 puts ""
136 end
137 end
138 end
138,1 Bot
Type here to search
ENG
4:03 PM
10/8/2020

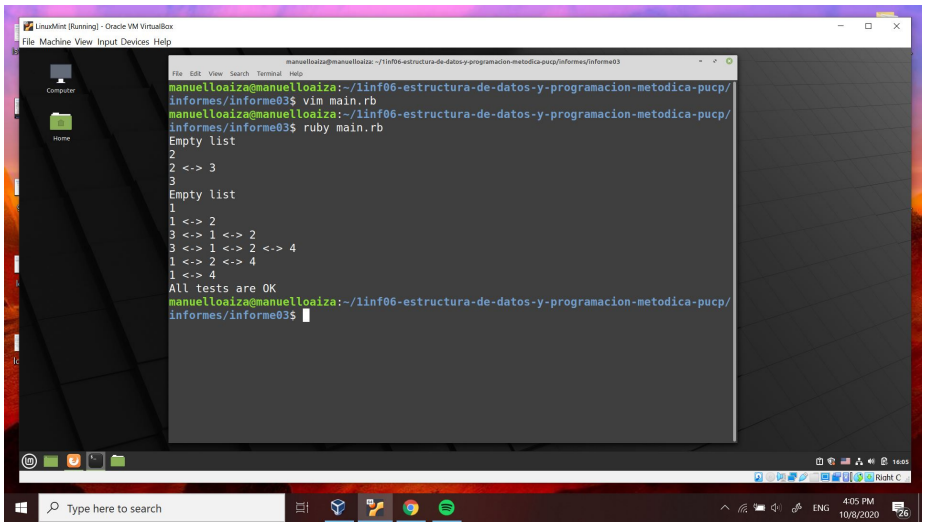
1.2 Testing


LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Computer
Home
File Edit View Search Terminal Help
manuelloaiza@manuelloaiza:~/1inf06-estructura-de-datos-y-programacion-metodica-pucp/infor
mes/informe03$ vim main.rb
Type here to search
ENG
4:03 PM
10/8/2020


LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
require_relative "./DoubleLinkedList.rb"
list1 = DoubleLinkedList.new
raise "Error in IsEmpty" unless list1.IsEmpty
list1.Print
list1.AddRight 2
list1.Print
list1.AddRight 3
list1.Print
raise "Error in AddRight" unless list1.Size == 2
raise "Error in Head" unless list1.Head == 2
list1.RemoveLeft
list1.Print
raise "Error in RemoveLeft" unless list1.Size == 1
raise "Error in Head" unless list1.Head == 3
list1.RemoveRight
list1.Print
raise "Error in RemoveRight" unless list1.IsEmpty
list1.AddLeft 1
list1.Print
list1.AddRight 2
list1.Print
list1.AddLeft 3
1,1 Top
Type here to search
ENG
4:04 PM
10/8/2020

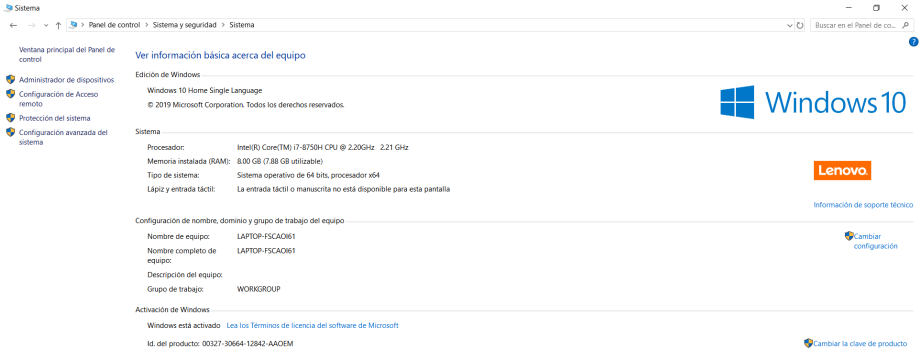

LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
raise "Error in Head" unless list1.Head == 2
list1.RemoveLeft
list1.Print
raise "Error in RemoveLeft" unless list1.Size == 1
raise "Error in Head" unless list1.Head == 3
list1.RemoveRight
list1.Print
raise "Error in RemoveRight" unless list1.IsEmpty
list1.AddLeft 1
list1.Print
list1.AddRight 2
list1.Print
list1.AddLeft 3
list1.Print
list1.AddRight 4
list1.Print
list1.Remove 3
list1.Print
raise "Error in Remove" unless list1.Size == 3
list1.Remove 2
list1.Print
raise "Error in Remove" unless list1.Size == 2
puts "All tests are OK"
38,1 Bot
Type here to search
ENG
4:04 PM
10/8/2020

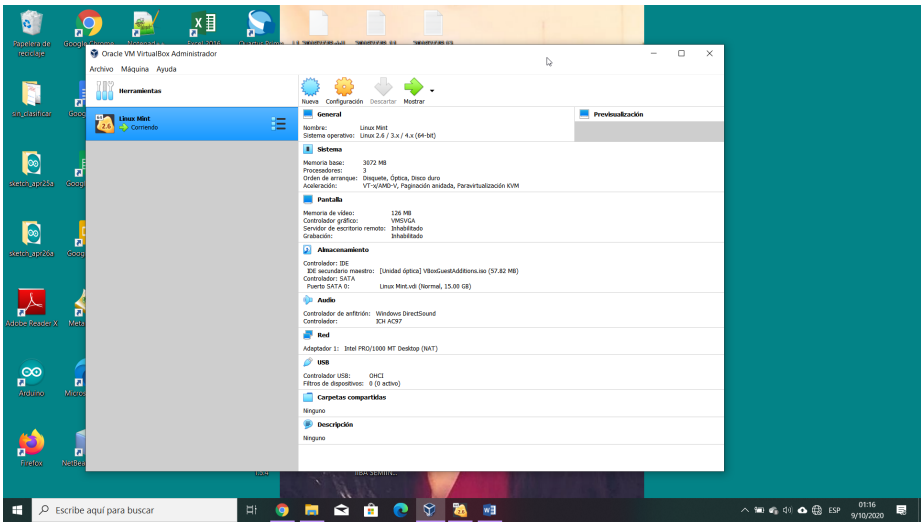

LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
manuelloaiza@manuelloaiza: ~/1inf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03
File Edit View Search Terminal Help
manuelloaiza@manuelloaiza:~/1inf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03$ vim main.rb
manuelloaiza@manuelloaiza:~/1inf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03$ ruby main.rb
Computer
Home
Type here to search
ENG
4:05 PM
10/8/2020

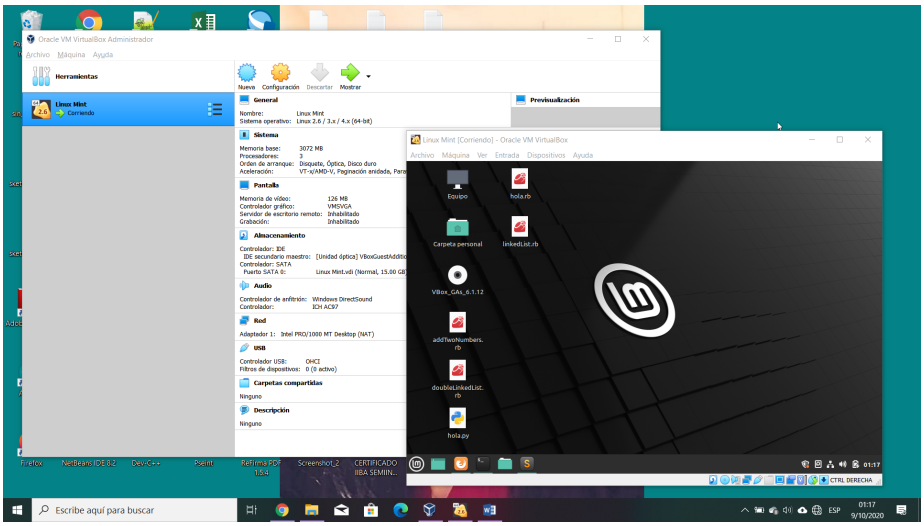

LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
manuelloaiza@manuelloaiza: ~/1inf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03
File Edit View Search Terminal Help
manuelloaiza@manuelloaiza:~/1inf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03$ vim main.rb
manuelloaiza@manuelloaiza:~/1inf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03$ ruby main.rb
Empty list
2
2 <-> 3
3
Empty list
1
1 <-> 2
3 <-> 1 <-> 2
3 <-> 1 <-> 2 <-> 4
1 <-> 2 <-> 4
1 <-> 4
All tests are OK
manuelloaiza@manuelloaiza:~/1inf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03$
Computer
Home
Type here to search
ENG
4:05 PM
10/8/2020

2 Oyarce Tocto, Elizabeth Patricia

2.1 Resources and Oracle VM Virtual Box parameters







2.2 Language research: Object Oriented Programming in Ruby

Ruby is an object-oriented programming language, that is, it meets the features of an OOP language.

1. Data Encapsulation
2. Data Abstraction
3. Polymorphism
4. Inheritance

Like all object-oriented programming languanges, Ruby involves classes and objects. A class is the abstraction of an object, it defines the behavior that objects will have and for them, it defines attributes, methods, fields and events. The class allows individual objects to be instantiated, therefore a class can be defined as a combination of characteristics and functions.

A class in Ruby always starts with the `class` keyword followed by the class name and finished with the keyword `end`, it is recommended to use camel case. The name must always be in initial capital letters. To exemplify, the class Person can be displayed as:

```
1  class Person
2  end
```

Ruby provides four types of variables:

1. Local Variables, they are the variables that are defined in a method. Local variables are not available outside the method and begin with a lowercase letter.

2. Instance Variables, are available in all methods or anywhere in the class. That means instance variables change from one object to another. Instance variables are preceded by the at sign `@` followed by the variable name.

3. Class Variables,are available in different objects. A class variable belongs to the class and to all the objects within it. They are preceded by the `@@` sign and followed by the variable name.

4. Global Variables,if you want to have a single variable, which is available in all classes, you must define a global variable. Global variables are always preceded by the dollar sign.

Methods are defined within a class as follows.

```
1  class Person
2
3      def initialize word
4          @name = word
5      end
6
7      def age value
8          @age = value
9      end
10 end
11
```

The initialize method is a special type of method (constructor), which will be executed when the `new` method of the class is called with the parameters.

```
me = Person.new "Elizabeth"
```

It should be noted that instance variables cannot be modified or obtained from outside the object. Below, we have an example to access and print the name of a person within the class.

```ruby
class Person

    def initialize word
        @name = word
        puts @name
    end

    def age value
        @age = value
    end
end


me = Person.new "Elizabeth"
```

```
elizabeth@Elizabeth-VirtualBox: ~/Escritorio
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
elizabeth@Elizabeth-VirtualBox:~$ cd Escritorio
elizabeth@Elizabeth-VirtualBox:~/Escritorio$ ruby conceptos.rb

Traceback (most recent call last):
conceptos.rb:14:in `<main>': undefined local variable or method `person' for mai
n:Object (NameError)
elizabeth@Elizabeth-VirtualBox:~/Escritorio$
elizabeth@Elizabeth-VirtualBox:~/Escritorio$ ruby conceptos.rb
Elizabeth
elizabeth@Elizabeth-VirtualBox:~/Escritorio$
```

Accessor methods (getters and setters) are used to access a property, getters allow obtaining properties, and setters are used to assign or modify a property.

```ruby
class Person

    def initialize word
        @name = word
    end

    def age value
        @age = value
    end

    def name
        @name
    end

    def name=(name2)
        @name = name2
    end
end


me = Person.new "Elizabeth"
puts me.name
me.name="Paty"
puts me.name

```

```
elizabeth@Elizabeth-VirtualBox: ~/Escritorio
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
elizabeth@Elizabeth-VirtualBox:~/Escritorio$ ruby conceptos.rb
Elizabeth
Paty
elizabeth@Elizabeth-VirtualBox:~/Escritorio$
```

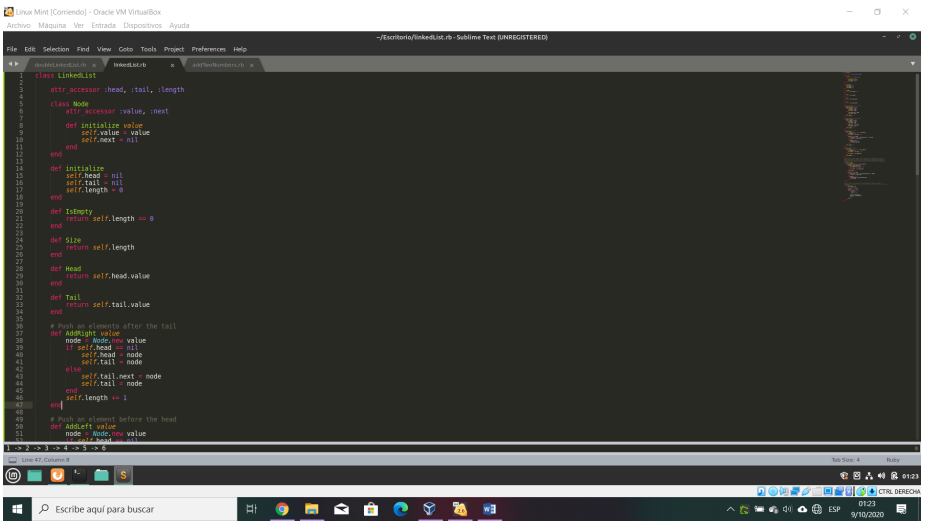
An easier way to access these properties is by defining attributes.

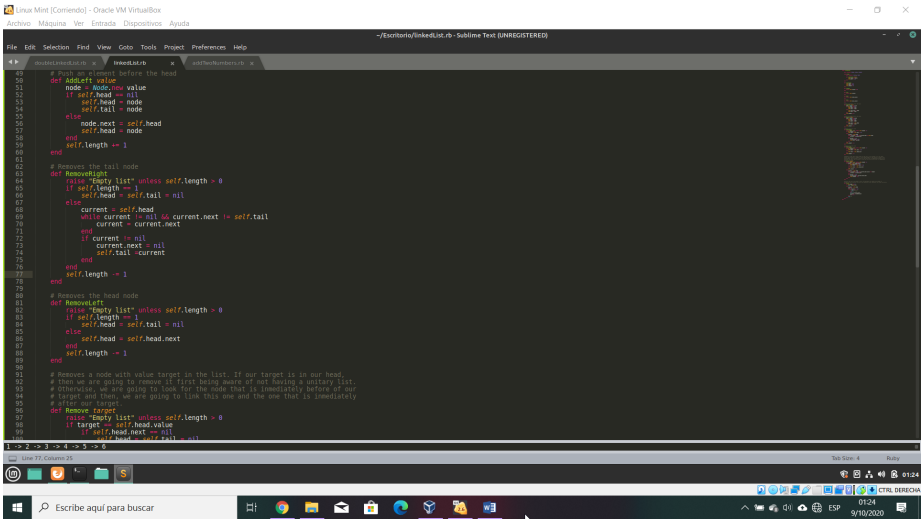
1. `attr_accessor` : getter and setter
2. `attr_reader` : getter
3. `attr_writer` : setter

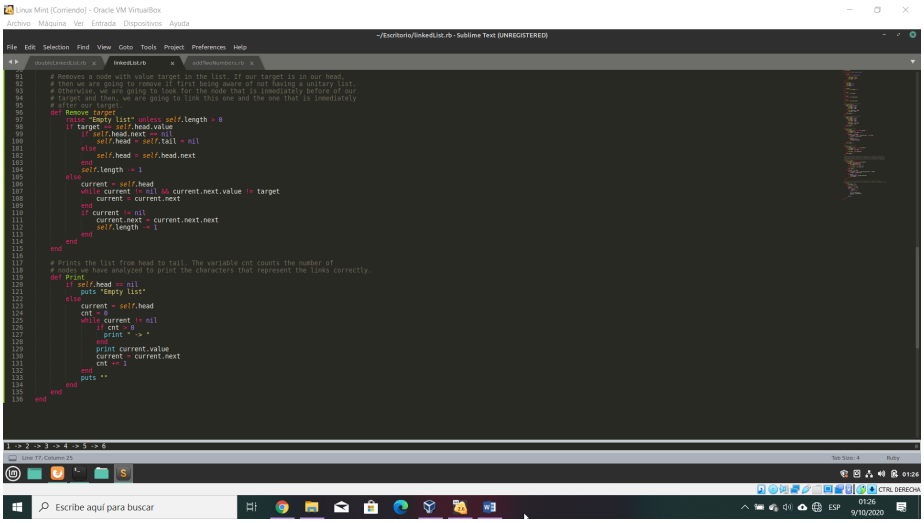
```ruby
class Person

    attr_accessor :name , :age

    def initialize (word)
        self.name = word
    end


end

me = Person.new "Elizabeth"
puts me.name
me.name ="Paty"
puts me.name
```

```
elizabeth@Elizabeth-VirtualBox: ~/Escritorio
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
elizabeth@Elizabeth-VirtualBox:~/Escritorio$ ruby conceptos.rb
Elizabeth
Paty
elizabeth@Elizabeth-VirtualBox:~/Escritorio$
```

Finally, an application of the classes is shown, using the node and list classes, which allow me to concatenate the nodes together to form a simple linked list, some methods are also defined that allow manipulating the nodes. This list will help us to develop the exercise that follows.

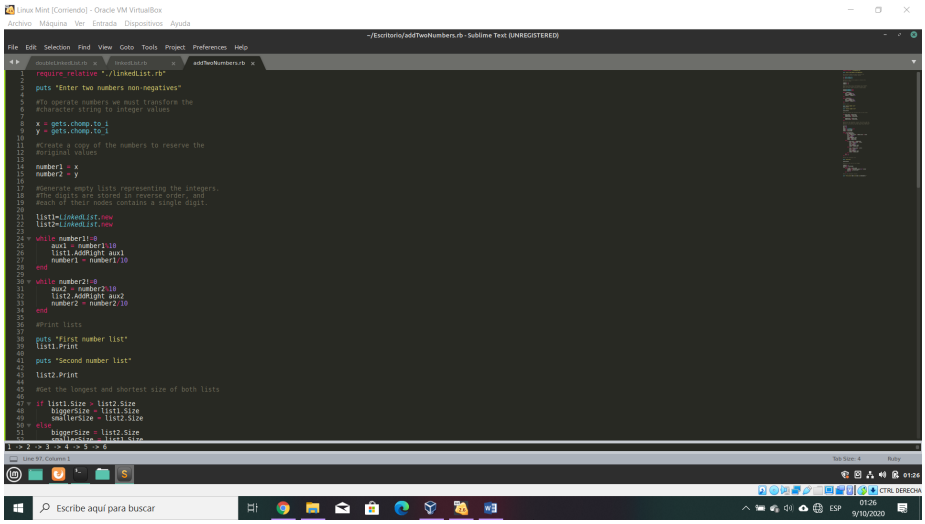


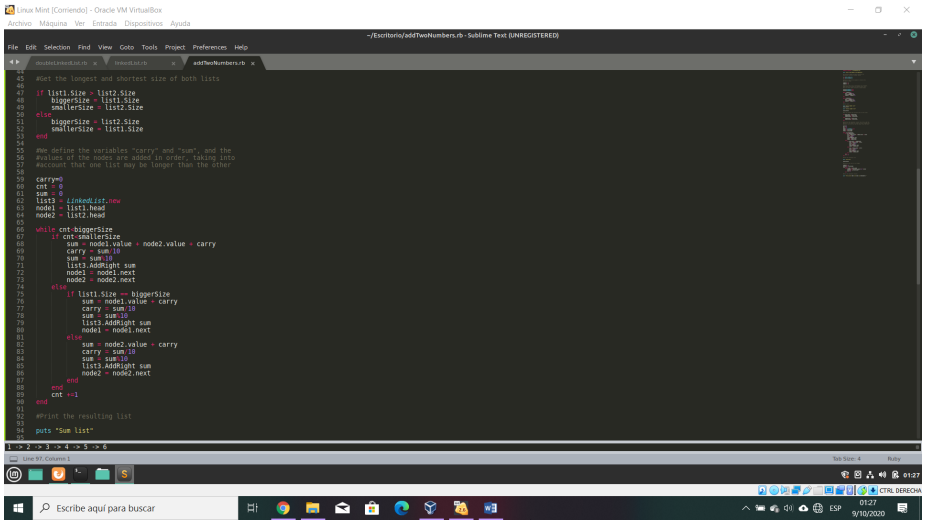


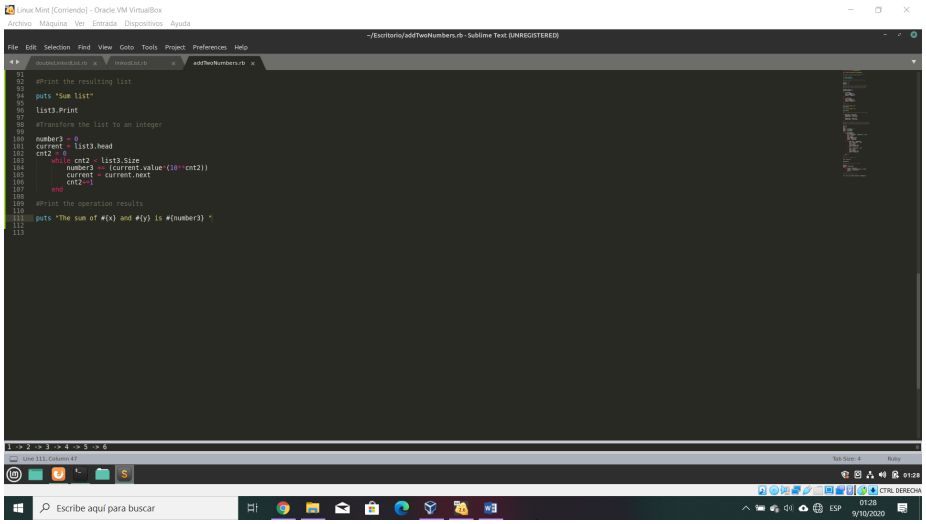


2.3 Example problem: Add Two Numbers

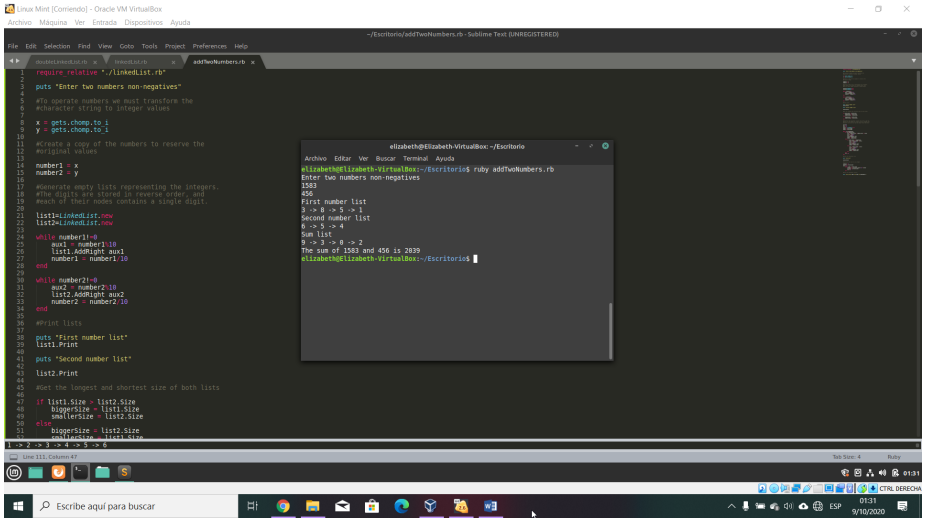
You are given two non-negative integers that you must represent in two non-empty linked lists. The digits are stored in reverse order and each of its nodes contains a single digit. Add the two numbers, return the sum and its linked list. You can assume that the two numbers do not contain any leading zeros, except for the number 0 itself.


Linux Mint [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
~/Escritorio/addTwoNumbers.rb - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
addTwoNumbers.rb
require_relative "./linkedList.rb"
puts "Enter two numbers non-negatives"
#To operate numbers we must transform the
#character string to integer values
x = gets.chomp.to_i
y = gets.chomp.to_i
#Create a copy of the numbers to reserve the
#original values
number1 = x
number2 = y
#Generate empty lists representing the integers.
#The digits are stored in reverse order, and
#each of their nodes contains a single digit.
list1=LinkedList.new
list2=LinkedList.new
while number1!=0
aux1 = number1%10
list1.AddRight aux1
number1 = number1/10
end
while number2!=0
aux2 = number2%10
list2.AddRight aux2
number2 = number2/10
end
#Print lists
puts "First number list"
list1.Print
puts "Second number list"
list2.Print
#Get the longest and shortest size of both lists
if list1.Size > list2.Size
biggerSize = list1.Size
smallerSize = list2.Size
else
biggerSize = list2.Size
1 -> 2 -> 3 -> 4 -> 5 -> 6
Line 97, Column 1
Tab Size: 4
Ruby
Escribe aquí para buscar
ESP
01:26
9/10/2020
CTRL DERECHA


Linux Mint [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
~/Escritorio/addTwoNumbers.rb - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
addTwoNumbers.rb
#Get the longest and shortest size of both lists
if list1.Size > list2.Size
biggerSize = list1.Size
smallerSize = list2.Size
else
biggerSize = list2.Size
smallerSize = list1.Size
end
#We define the variables "carry" and "sum", and the
#values of the nodes are added in order, taking into
#account that one list may be longer than the other
carry=0
cnt = 0
sum = 0
list3 = LinkedList.new
node1 = list1.head
node2 = list2.head
while cnt<biggerSize
if cnt<smallerSize
sum = node1.value + node2.value + carry
carry = sum/10
sum = sum%10
list3.AddRight sum
node1 = node1.next
node2 = node2.next
else
if list1.Size == biggerSize
sum = node1.value + carry
carry = sum/10
sum = sum%10
list3.AddRight sum
node1 = node1.next
else
sum = node2.value + carry
carry = sum/10
sum = sum%10
list3.AddRight sum
node2 = node2.next
end
end
cnt +=1
end
#Print the resulting list
puts "Sum list"
1 -> 2 -> 3 -> 4 -> 5 -> 6
Line 97, Column 1
Tab Size: 4
Ruby
Escribe aquí para buscar
ESP
01:27
9/10/2020
CTRL DERECHA

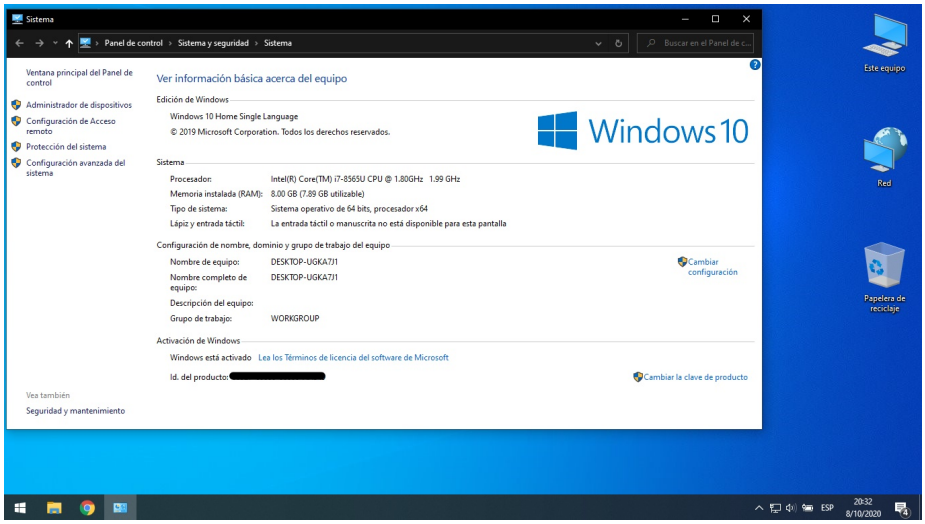


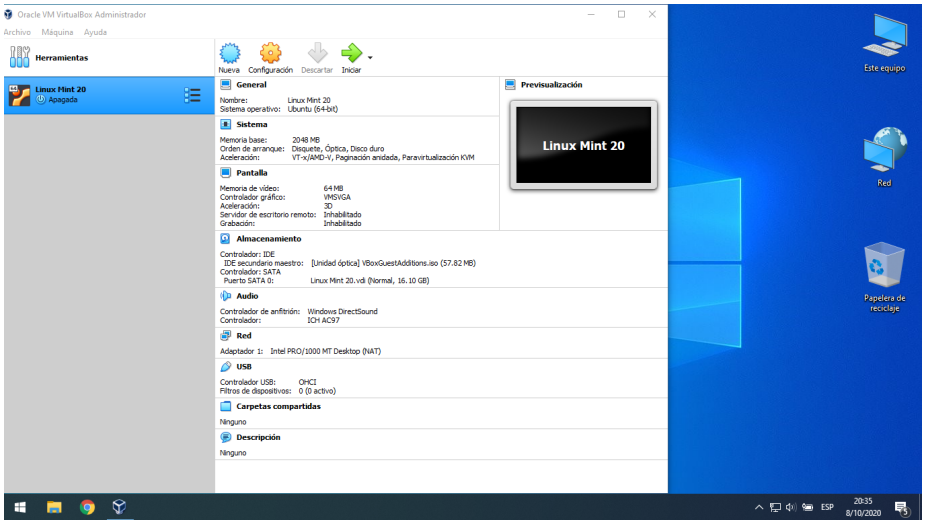
The following picture is from the program execution

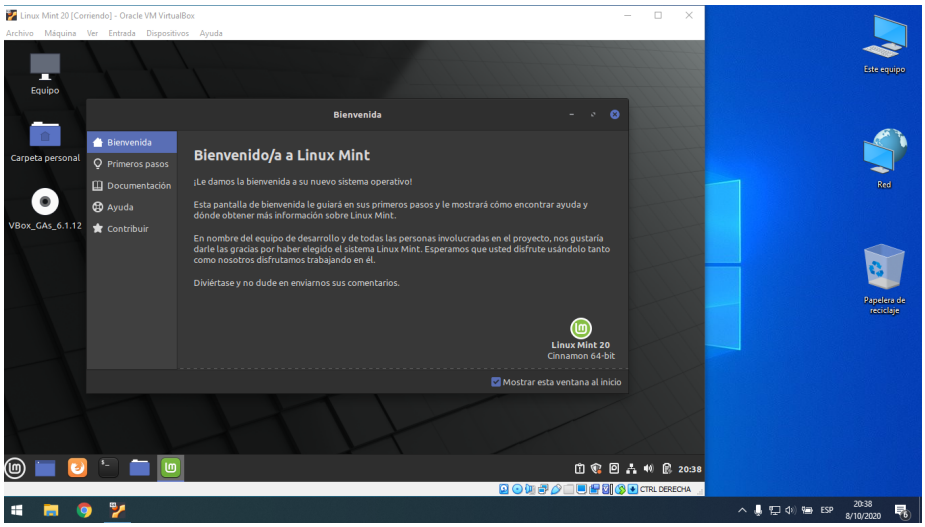


3 Saras Rivera, André Edgardo

3.1 Resources and Oracle VM Virtual Box parameters

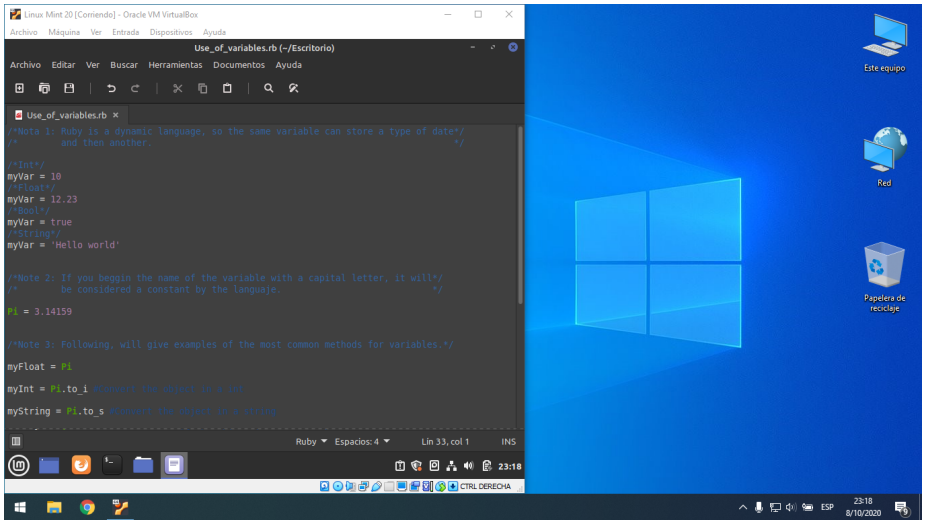


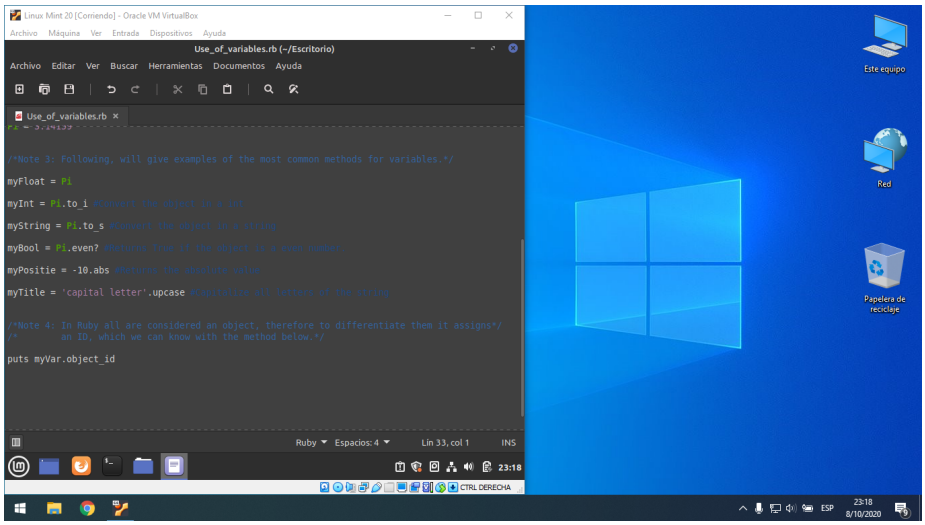


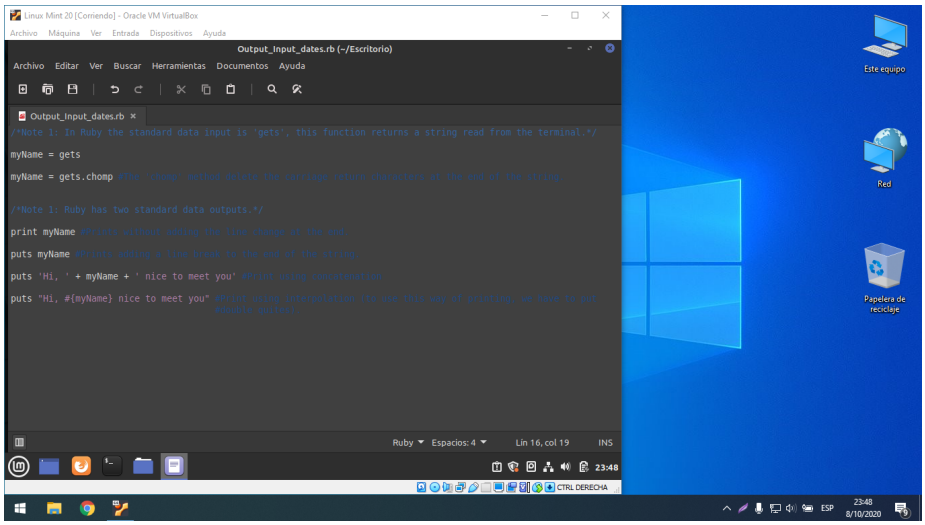


3.2 Language research: Ruby Basic Syntax and Commands

First of all, it should be clarified Ruby is a dynamic language where all variables including constants are treated as objects, that is to say, operations performed at compilation time occur at runtime, it is not necessary to declare variables either. In this context, arithmetic, comparison and ternary operators will be considered methods. On the following pages, will be explained using examples below:



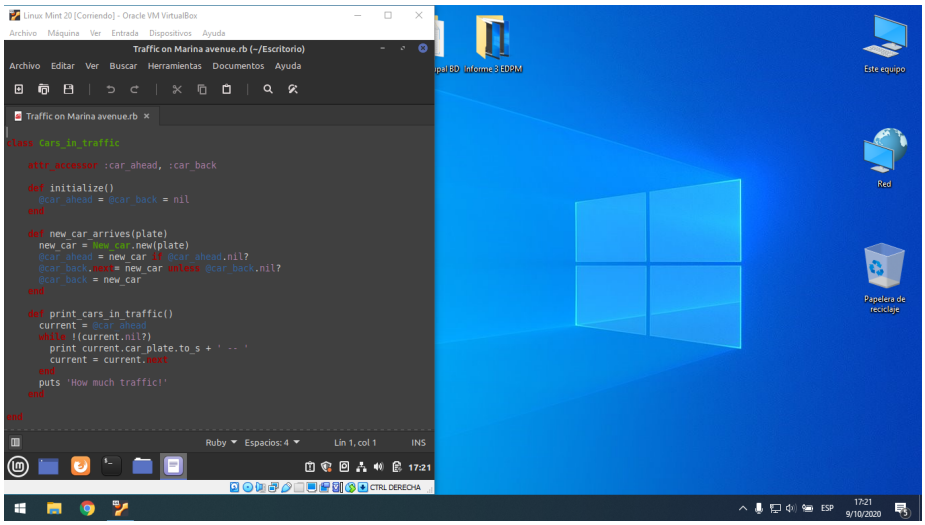


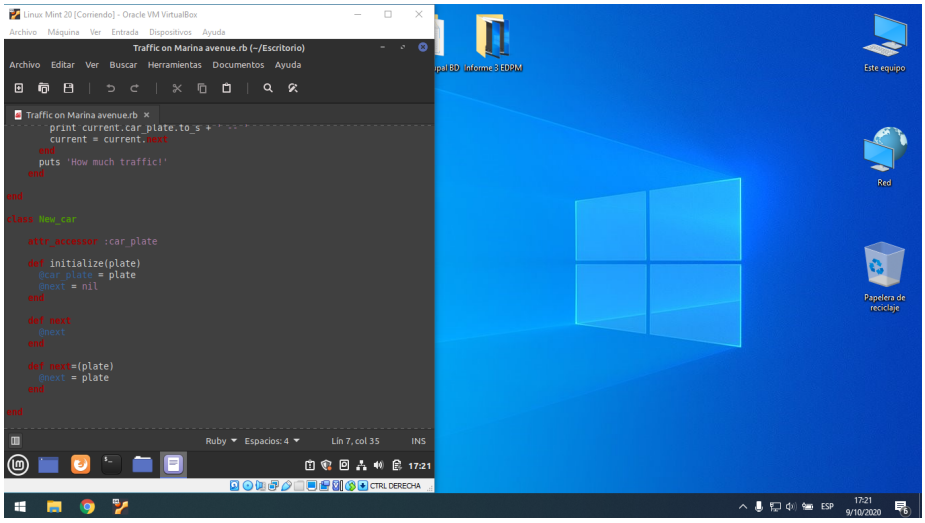


Linked structures such as lists, stacks and queues are not created by default in Ruby, however, the arrays defined in this language are dynamic (the size can be varied). In that context, we can implement linked structures through classes and create methods to handle them as seen in the introductory example. The concepts of what a linked structure is already found in the Data Structures and Algorithms Using Python book provided by the teacher in the course bibliography, for that reason, it will not go into much depth on this topic.

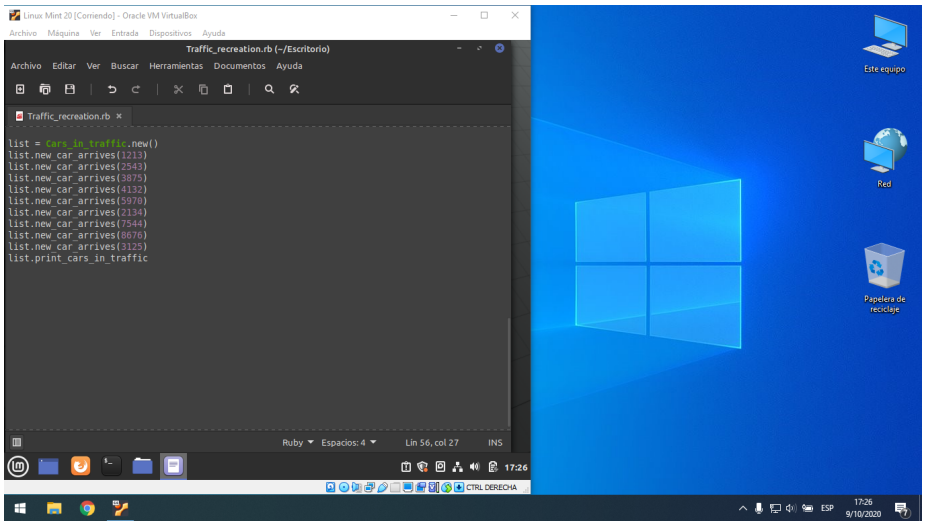
3.3 Example problem: Traffic Recreation

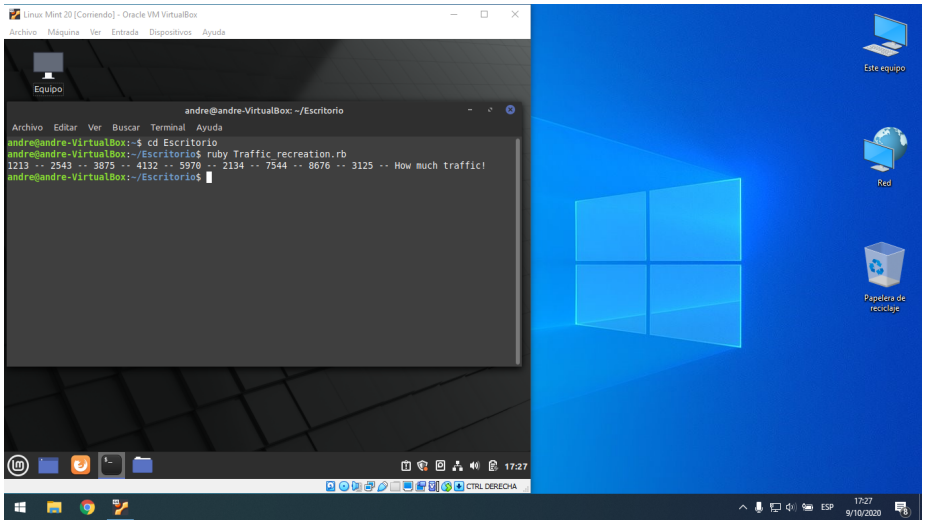
The use of the concepts mentioned above will be explained by means of a simple and didactic example, in this case the traffic simulation using a queue.





First of all, we define our class Cars in traffic which simulates a group of cars stalled in traffic and we can differentiate them through their Plate. Inside our first defined class, we will mainly create three methods: The first one, initializes the variables defined by the access method attr accessor to null value. The second, adds one more car to traffic, identifying it by means of its license plate. It also checks if the queue is empty (in null value) and if that were the case, it increases the cart to said empty queue. The third, prints all the cars in traffic by means of a 'while' that ends when the head reaches the null value. Within our second defined class, we will only create the node New car by initializing it and defining its variables using attr accessor and def.

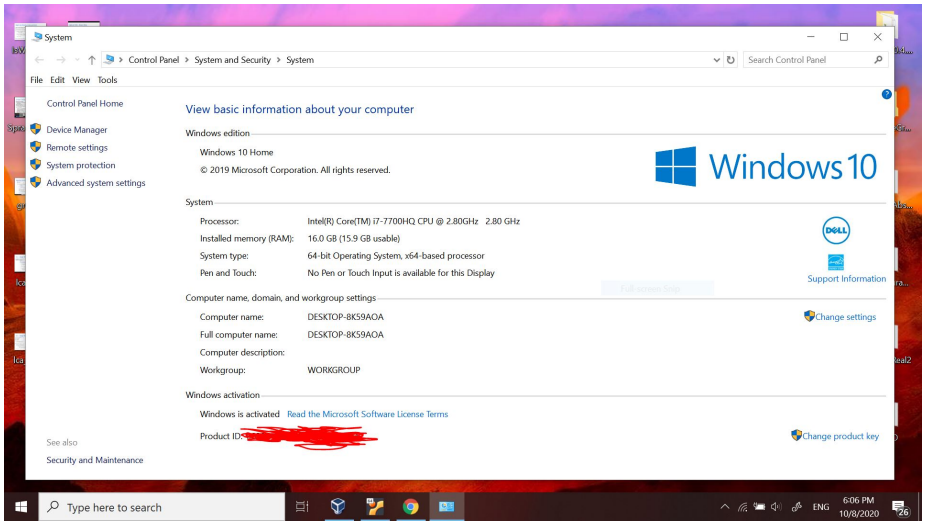


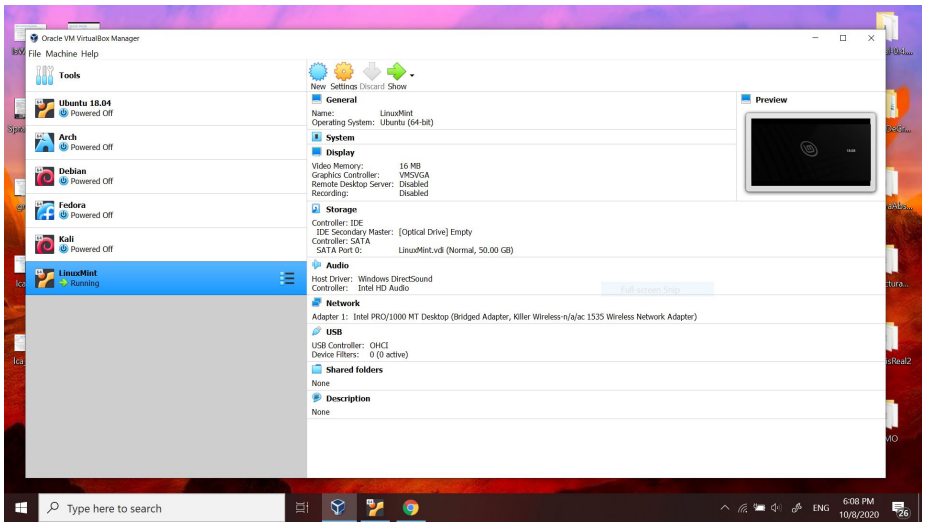


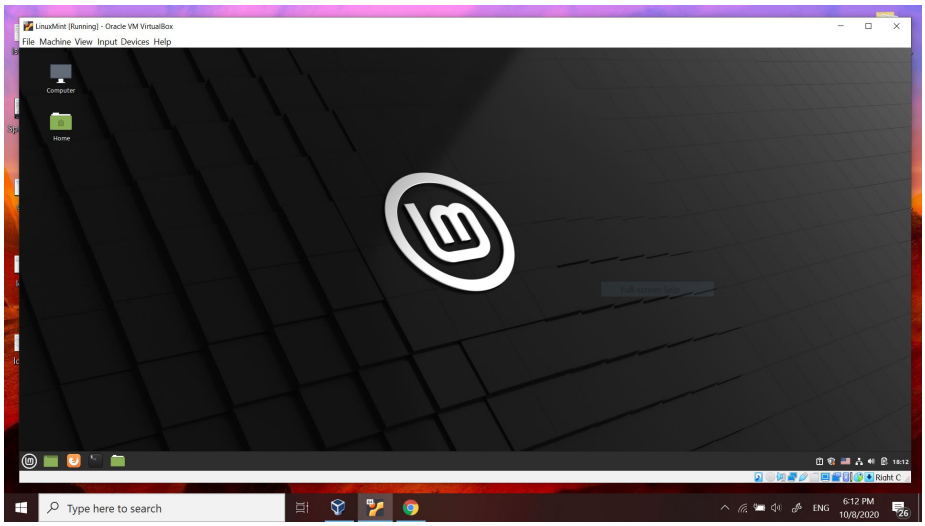
In the example shown, first the class 'Cars in Traffic' is instantiated in a variable named for this case 'list' (list of cars stuck in traffic), consequently we proceed to increase values to the queue one by one to finally print all cars.

4 Loaiza Vasquez, Manuel Alejandro

4.1 Resources and Oracle VM Virtual Box parameters

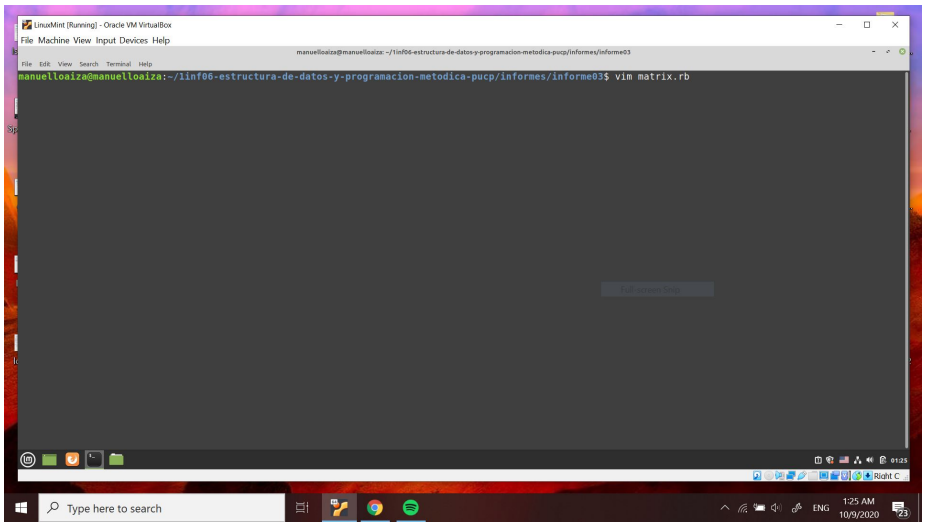


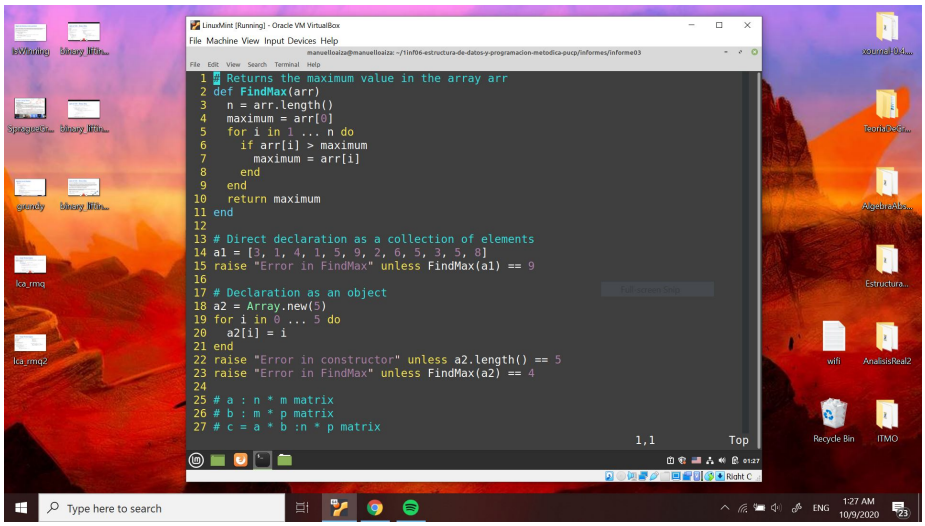


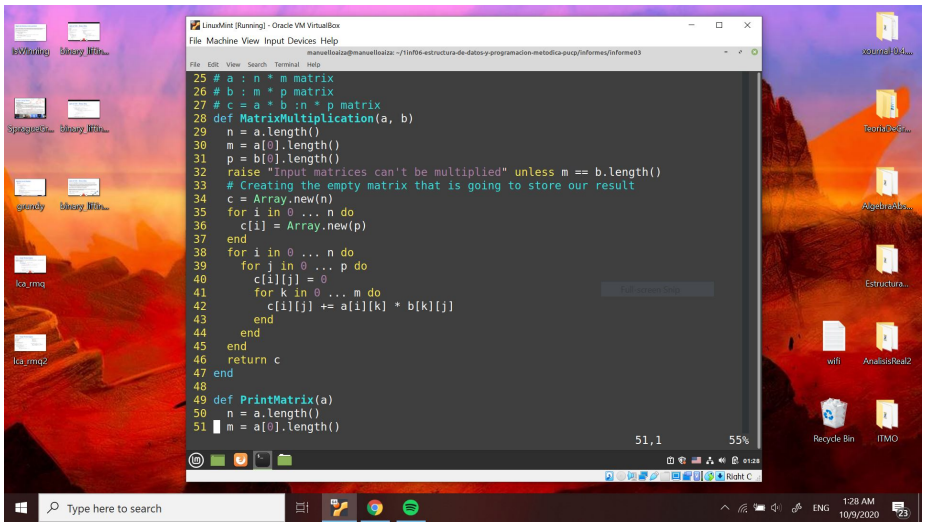


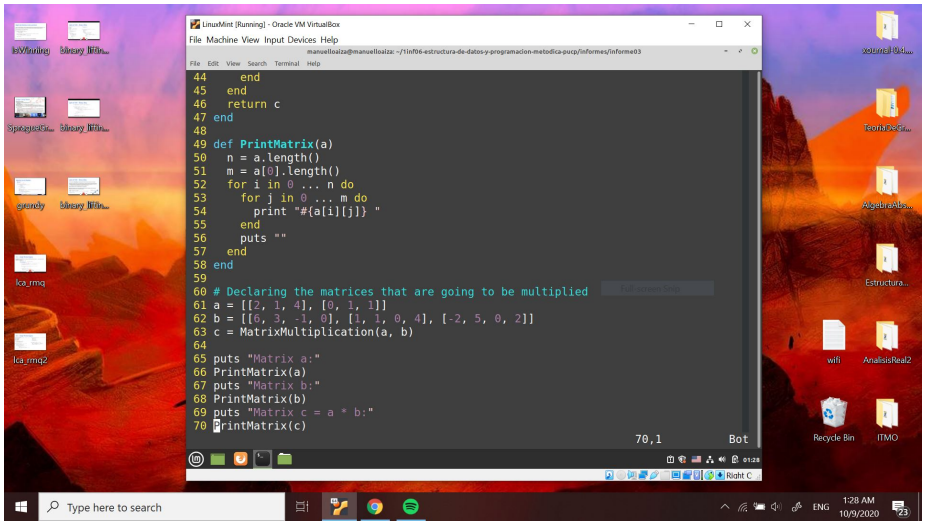
4.2 Language research: Arrays, strings, exceptions and files

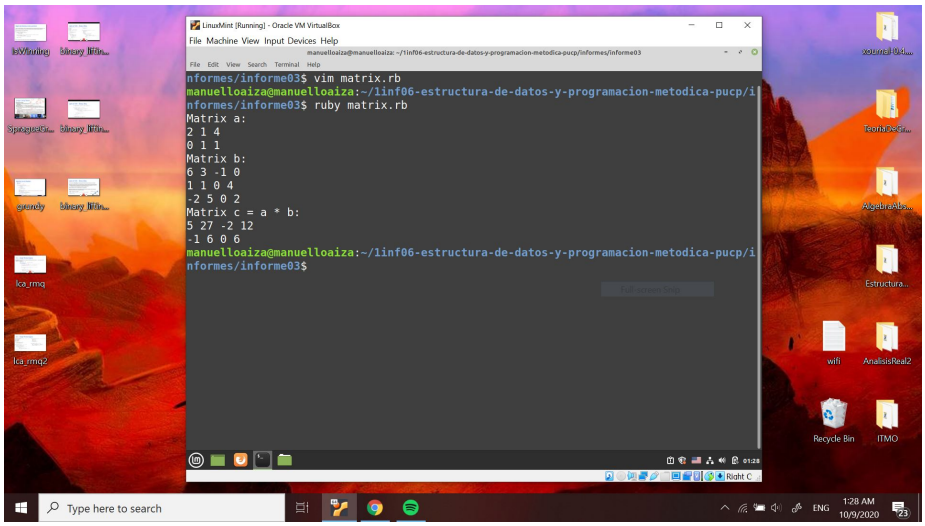
For the adjacency matrix representation of a graph we need a 2D array and for the adjacency list representation of a graph we need a one dimensional array, so it was very important to do some research about them. Arrays in Ruby are instances of a class called `Array` which has methods such as `length`, many constructors that can receive one, many or no elements and we can even give the constructor the number of elements and which elements are going to be there by default. We can see how arrays work in my example on section 4.3 in which I have used an array of lists to iterate over them to obtain the sorted list and other arrays to store information about positions and length to decrease the time complexity of the algorithm. In the following example, I am going to declare arrays in different ways, iterate through them via indices with a `for` loop, and also creating arrays of arrays, also know as matrices, which I am going to print and multiply using some functions that I will define.


LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
manuelloaiza@manuelloaiza:~/1inf06-estructura-de-datos-y-programacion-metodica-pucp/informes/informe03$ vim matrix.rb
Type here to search
ENG
1:25 AM
10/9/2020

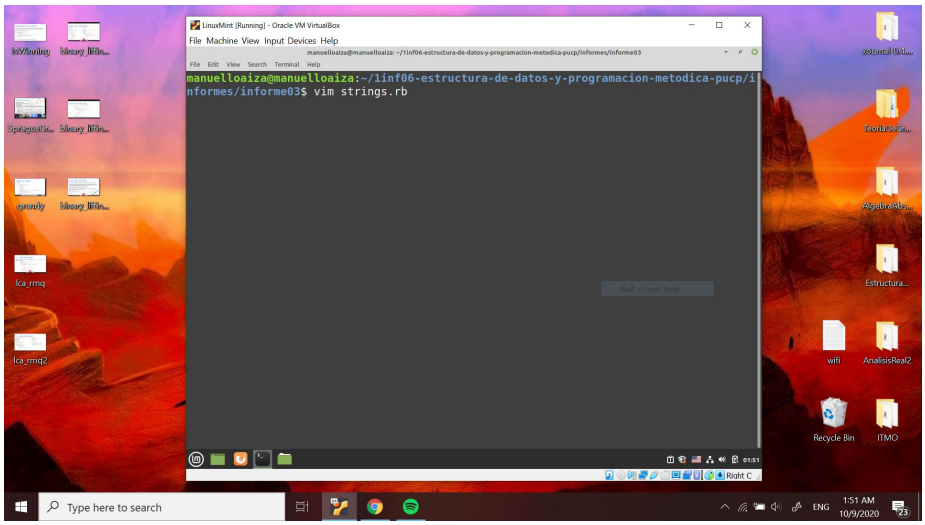

LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Returns the maximum value in the array arr
def FindMax(arr)
n = arr.length()
maximum = arr[0]
for i in 1 ... n do
if arr[i] > maximum
maximum = arr[i]
end
end
return maximum
end
# Direct declaration as a collection of elements
a1 = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8]
raise "Error in FindMax" unless FindMax(a1) == 9
# Declaration as an object
a2 = Array.new(5)
for i in 0 ... 5 do
a2[i] = i
end
raise "Error in constructor" unless a2.length() == 5
raise "Error in FindMax" unless FindMax(a2) == 4
a : n * m matrix
b : m * p matrix
c = a * b :n * p matrix
1,1 Top
Type here to search
ENG
1:27 AM
10/9/2020

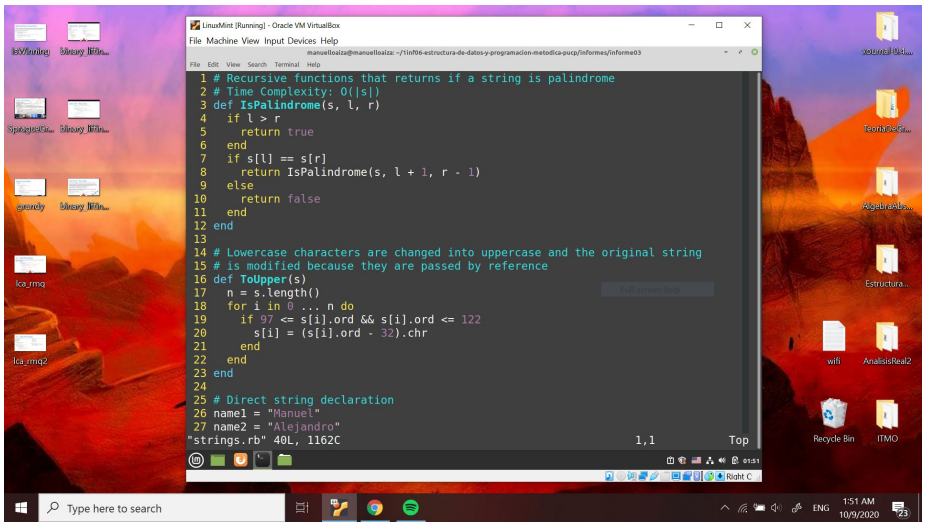

LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
25 # a : n * m matrix
26 # b : m * p matrix
27 # c = a * b :n * p matrix
28 def MatrixMultiplication(a, b)
29 n = a.length()
30 m = a[0].length()
31 p = b[0].length()
32 raise "Input matrices can't be multiplied" unless m == b.length()
33 # Creating the empty matrix that is going to store our result
34 c = Array.new(n)
35 for i in 0 ... n do
36 c[i] = Array.new(p)
37 end
38 for i in 0 ... n do
39 for j in 0 ... p do
40 c[i][j] = 0
41 for k in 0 ... m do
42 c[i][j] += a[i][k] * b[k][j]
43 end
44 end
45 end
46 return c
47 end
48
49 def PrintMatrix(a)
50 n = a.length()
51 m = a[0].length()
51,1 55%
Type here to search
ENG
1:28 AM
10/9/2020

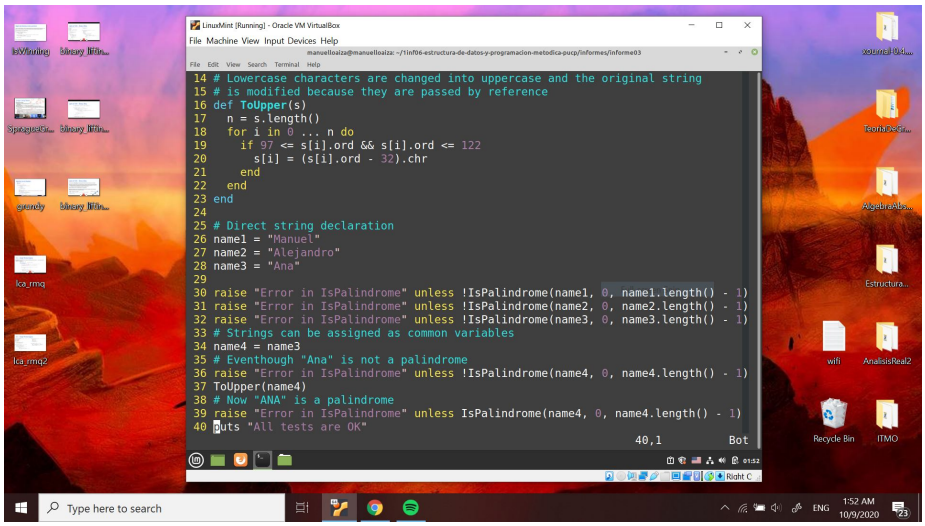

LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
44 end
45 end
46 return c
47 end
48
49 def PrintMatrix(a)
50 n = a.length()
51 m = a[0].length()
52 for i in 0 ... n do
53 for j in 0 ... m do
54 print "#{a[i][j]} "
55 end
56 puts ""
57 end
58 end
59
60 # Declaring the matrices that are going to be multiplied
61 a = [[2, 1, 4], [0, 1, 1]]
62 b = [[6, 3, -1, 0], [1, 1, 0, 4], [-2, 5, 0, 2]]
63 c = MatrixMultiplication(a, b)
64
65 puts "Matrix a:"
66 PrintMatrix(a)
67 puts "Matrix b:"
68 PrintMatrix(b)
69 puts "Matrix c = a * b:"
70 PrintMatrix(c)
70,1 Bot
Type here to search
ENG
1:28 AM
10/9/2020

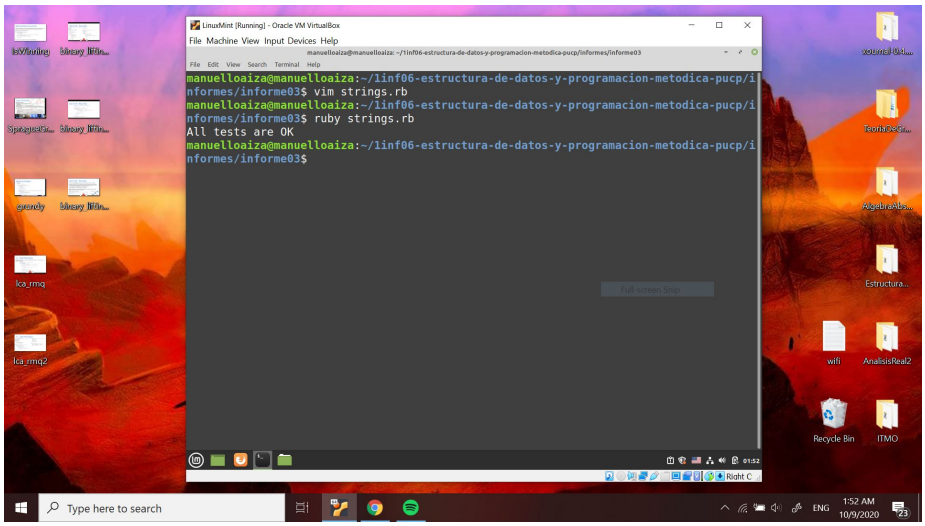


In Ruby, we have the class `String`, a collection of elements of one bite, they usually represent the characters from 0 to 255. In C, it was very easy to do arithmetic operations with them. In Ruby, we can use the methods `ord` and `chr` to obtain the ASCII code of a character and to convert the ASCII code to the character, respectively. Then, as other containers, we can iterate through them via indeces, concatenate strings using + operator, obtain their size using `length`, parse them and much more. In the following program, I am going to test that using some functions to check if a string is palindrome, and converting lowercase characters into uppercase because comparison is case sensitive.




LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Recursive functions that returns if a string is palindrome
Time Complexity: O(|s|)
def IsPalindrome(s, l, r)
if l > r
return true
end
if s[l] == s[r]
return IsPalindrome(s, l + 1, r - 1)
else
return false
end
end
Lowercase characters are changed into uppercase and the original string
is modified because they are passed by reference
def ToUpper(s)
n = s.length()
for i in 0 ... n do
if 97 <= s[i].ord && s[i].ord <= 122
s[i] = (s[i].ord - 32).chr
end
end
end
# Direct string declaration
name1 = "Manuel"
name2 = "Alejandro"
"strings.rb" 40L, 1162C
1,1 Top
Type here to search
ENG
1:51 AM
10/9/2020


LinuxMint [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Lowercase characters are changed into uppercase and the original string
is modified because they are passed by reference
def ToUpper(s)
n = s.length()
for i in 0 ... n do
if 97 <= s[i].ord && s[i].ord <= 122
s[i] = (s[i].ord - 32).chr
end
end
end
# Direct string declaration
name1 = "Manuel"
name2 = "Alejandro"
name3 = "Ana"
raise "Error in IsPalindrome" unless !IsPalindrome(name1, 0, name1.length() - 1)
raise "Error in IsPalindrome" unless !IsPalindrome(name2, 0, name2.length() - 1)
raise "Error in IsPalindrome" unless !IsPalindrome(name3, 0, name3.length() - 1)
# Strings can be assigned as common variables
name4 = name3
# Eventhough "Ana" is not a palindrome
raise "Error in IsPalindrome" unless !IsPalindrome(name4, 0, name4.length() - 1)
ToUpper(name4)
# Now "ANA" is a palindrome
raise "Error in IsPalindrome" unless IsPalindrome(name4, 0, name4.length() - 1)
puts "All tests are OK"
40,1 Bot
Type here to search
ENG
1:52 AM
10/9/2020

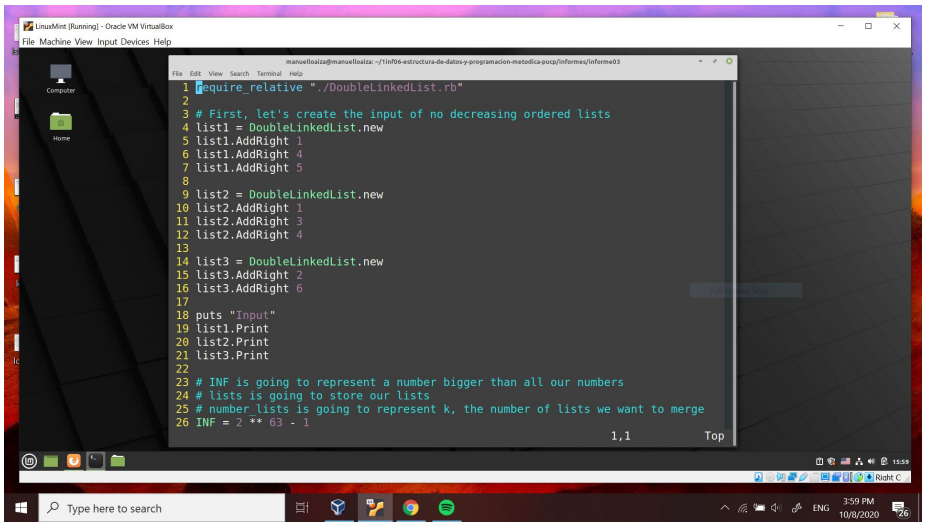


To handle exceptions, we can use `raise ... unless ...` as we use `assert` in C or Python. We can find other ways to use assertions by creating modules and including them in our class definition. I have used raise in our testing code on section 1.2 to check if our double linked list methods are correct and in the previous examples with arrays and strings.

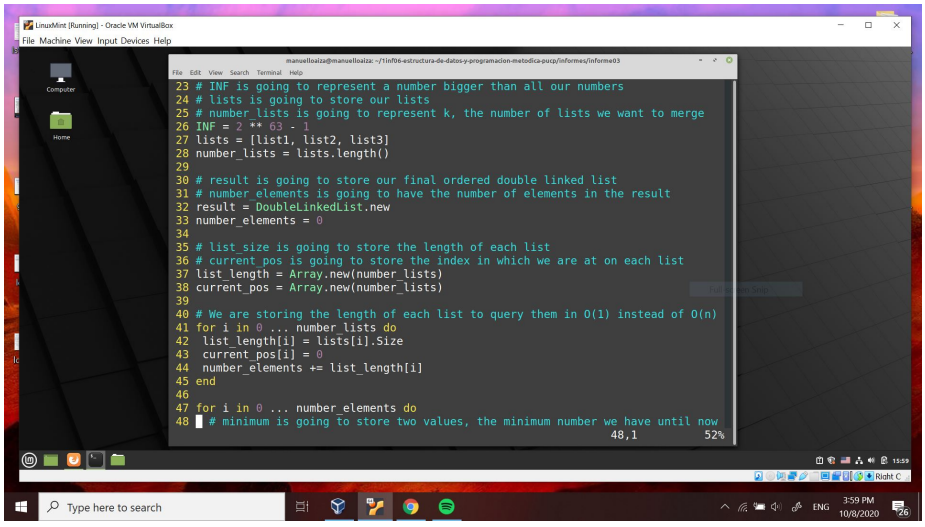
When we want to load files from other sources, we can use `require` and write the string that with the path of the file we want to use. If that file is in the same directory of our current file, we can use `require_relative`. In my example below and in the testing, I have used this statement to load our double linked list class definition from `DoubleLinkedList.rb` file.

4.3 Example problem: Merged k Sorted Lists

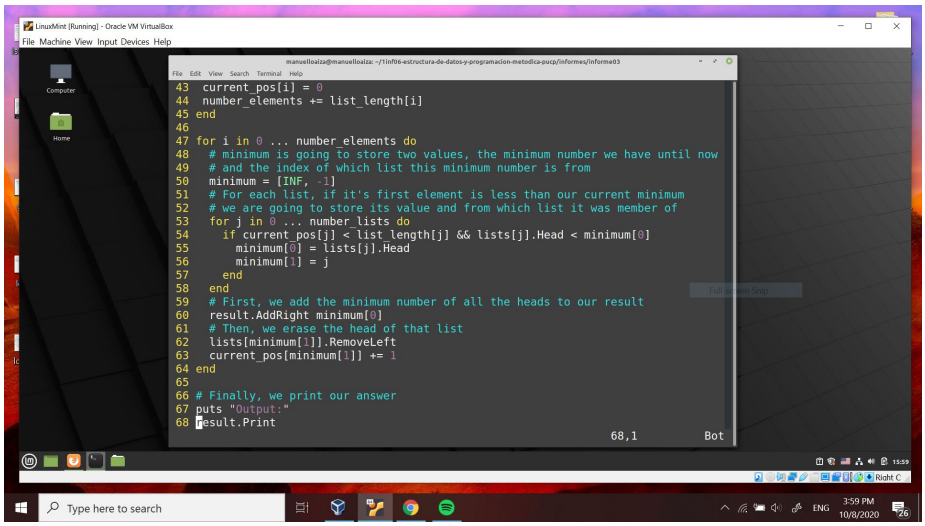
You are given an array of k double linked lists, each double linked list is sorted in non decreasing order. Merge all the double linked lists into one double linked list and return it.

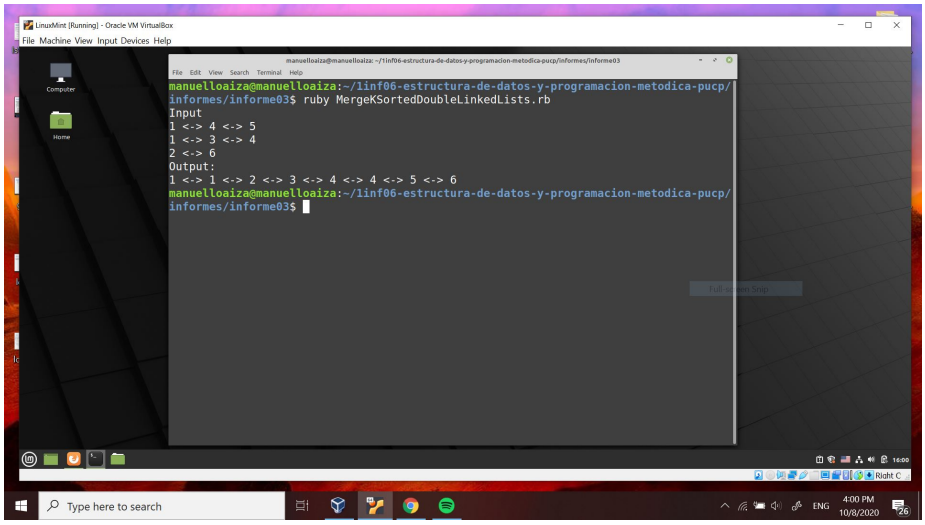


```ruby
require_relative "./DoubleLinkedList.rb"

# First, let's create the input of no decreasing ordered lists
list1 = DoubleLinkedList.new
list1.AddRight 1
list1.AddRight 4
list1.AddRight 5

list2 = DoubleLinkedList.new
list2.AddRight 1
list2.AddRight 3
list2.AddRight 4

list3 = DoubleLinkedList.new
list3.AddRight 2
list3.AddRight 6

puts "Input"
list1.Print
list2.Print
list3.Print

# INF is going to represent a number bigger than all our numbers
# lists is going to store our lists
# number_lists is going to represent k, the number of lists we want to merge
INF = 2 ** 63 - 1
```



```ruby
# INF is going to represent a number bigger than all our numbers
# lists is going to store our lists
# number_lists is going to represent k, the number of lists we want to merge
INF = 2 ** 63 - 1
lists = [list1, list2, list3]
number_lists = lists.length()

# result is going to store our final ordered double linked list
# number_elements is going to have the number of elements in the result
result = DoubleLinkedList.new
number_elements = 0

# list_size is going to store the length of each list
# current_pos is going to store the index in which we are at on each list
list_length = Array.new(number_lists)
current_pos = Array.new(number_lists)

# We are storing the length of each list to query them in O(1) instead of O(n)
for i in 0 ... number_lists do
 list_length[i] = lists[i].Size
 current_pos[i] = 0
 number_elements += list_length[i]
end

for i in 0 ... number_elements do
 # minimum is going to store two values, the minimum number we have until now
```





References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, Cambridge, Massachussets, 2009.

[2] Tutorialspoint: Ruby Basics, `https://www.tutorialspoint.com/ruby/index.htm`

[3] Ruby-docs.org: IO, `https://ruby-doc.org/core-2.7.2/IO.html#method-i-puts`

[4] Stack Overflow: What is the difference between print and puts?, https://stackoverflow.com/questions/5018633/what-is-the-difference-between-print-and-puts

[5] Stack Overflow: Is it idiomatic Ruby to add an assert( ) method to Ruby's Kernel class?, https://stackoverflow.com/questions/147969/is-it-idiomatic-ruby-to-add-an-assert-method-to-rubys-kernel-class

[6] Ruby-docs.org: Arrays, https://ruby-doc.org/core-2.4.1/Array.html

[7] Ruby-docs.org: Strings, https://ruby-doc.org/core-2.4.0/String.html

[8] LeetCode: Reverse Linked List, https://leetcode.com/problems/reverse-linked-list/

[9] LeetCode: Add Two Numbers, https://leetcode.com/problems/add-two-numbers/

[10] LeetCode: Merge k Sorted Lists, https://leetcode.com/problems/merge-k-sorted-lists/