Lab 1: Introduction to Ruby

Learning Objectives:

1. Introduction to Ruby programming
2. Getting up and running with Ruby
3. Running your first ruby program
4. Basic Ruby syntaxes
   4.1. Strings
   4.2. Arrays and Hashes
   4.3. User Input
   4.4. If-else, Switch-case, Loops and Iterators
   4.5. Methods
   4.6. Classes and Objects
5. Introduction to Interactive Ruby
6. Introduction to RubyGems

Supporting files:

1. MyFirstProgram.rb
2. ArrayTest.rb
3. ECE421–Using Ruby Version Manager.pdf

---

1. Introduction to Ruby programming

Ruby is a dynamic, interpreted, open source, and a truly object oriented programming language. Ruby was created in Japan by Yukihiro Matsumoto (a.k.a 'Matz') in 1993. The latest Ruby version is 2.5.0 released on December 25, 2017. Ruby is a server side scripting language like Perl and Python, created with a focus on simplicity and productivity. Hence, the Ruby syntax is simple and high-level that is easy to understand and write.

Features of Ruby:

- A programmer friendly language.
- A dynamically typed server side scripting language.
- In Ruby, everything that can be assigned to a variable is an object.
- Is a powerful language with Blocks, Modules, and Lambdas.
- Supports arbitrary precision arithmetic.
- Is open-source but is subject to a license.
- Can be embedded in Hypertext Markup Language (HTML).
- Supports many GUI tools such as GTK, and OpenGL.
- Can easily be connected to DB2, MySQL, Oracle, and Sybase.

---

2. Getting up and running with Ruby

Usually Mac OS X and Unix environments come with default Ruby installations. You can check the installed version in the lab's computers by opening the Terminal and typing:

```
$ ruby -v
```

For Windows environments you can use this one step installer.
The complete Ruby installation guide for different needs and platforms can be found here.

---

3. Running your first Ruby program

Once Ruby is installed, in a Linux system you can create your own Ruby program by opening the Terminal and typing:

```
$ gedit <filename>.rb
```

The 'gedit' command opens a blank file with the entered file name and extension if the file is not found in the current working directory.
In order to execute a Ruby program, open the Terminal, move to the directory where the desired program is saved, enter the following command in the Terminal:

```
$ ruby <filename>.rb
```

Task 1: Execute the attached MyFirstProgram.rb file from your Terminal using the above mentioned process.

---

4. Basic Ruby syntaxes

This section gives a high-level overview of some basic Ruby syntaxes. Refer to the MyFirstProgram.rb file for detailed examples of each sub-sections.

Task 2: Open the MyFirstProgram.rb using a text editor and follow the examples of the following sub-sections. Play around and experiment with the code as you wish.

4.1 Strings

A String object in Ruby holds and manipulates an arbitrary sequence of one or more bytes, typically representing characters that represent human language. The text within the quotation marks is the value of the string. Strings in Ruby are objects, and unlike other languages, strings are mutable, which means they can be changed in places instead of creating new strings. For example:

```
$ str = "String"
$ puts "First three characters of the String:  #{str[0,3]}"
```

Output: First three characters of the String: Str

4.2 Arrays and Hashes

Arrays in Ruby are ordered, integer-indexed collections of any object. Each element in an array is associated with and referred to by an index. Array indexing starts at 0, as in C or Java. A negative index is assumed relative to the end of the array that is, an index of -1 indicates the last element of the array. Ruby arrays can hold objects such as String, Integer, Hash, Symbol, even other Array objects. For example, you can create an array using:

```
$ my_arr = [1,2,"name",4,true]
```

A **Hash** is a collection of key-value pairs such as: "key" = > "value". It is similar to an Array, except that indexing is done via arbitrary keys of any object type, not an integer index. For example:

```
$ numbers = Hash.new(0)
$ numbers = {“one” => 1, “two” => 2}
```

4.3 User Input

We use the **gets** method to get the user input as a String. We can add the **chomp** method along with gets in order to get rid of the trailing new-line character from the input. For example:

```
$ input = gets.chomp
```

4.4 If-else, Switch-case, Loops, and Iterators

Ruby offers conditional structures of if-else and switch-case as any other programming languages. Ruby offers **while**, **for**, and **until**, loops along with the **next**, **redo**, and **retry** statements. **Iterators** are methods that are supported by collections in Ruby. Iterators return all the elements of a collection, one by one, one element at a time. Syntaxes of some of these concepts are discussed in MyFirstProgram.rb file.

4.5 Methods

Ruby methods are very similar to functions in any other programming language. Ruby methods are used to bundle one or more repeatable statements into a single unit. Syntax of a Ruby method is as follows:

```
$ def <methodName> (<arguments>)
        …..code….
end
```

4.6 Classes and Objects

Like every other object oriented programming language, Ruby involves classes and objects. A class is a blueprint from which individual objects are created. Classes offer four types of variables in Ruby namely, **local**, **instance**, **class**, and **global** variables. A brief example of a class and its objects are discussed in MyFirstProgram.rb file

---

5. Introduction to Interactive Ruby

Interactive Ruby Shell or **‘irb’** is a Ruby Shell to interactively execute Ruby expressions read from the standard input. It is one of most popular Ruby features, especially with new developers. IRB is pretty handy for quickly trying something out, and it is a great tool for exploring the language Ruby, and things that come built in. You can bash out a one-liner, try a method you've just learned about, or even build a small algorithm or two without going the whole way to writing a complete program.

Task 3: To start the IRB shell, open Terminal and type the following:

```
$ irb
```

The command starts the IRB shell. Notice how the "prompt" indicator changes to '>'. Type the following code in the irb shell and play around with the code as you like.

```
irb(main) :001:0> 2 + 3
```

Output: => 5

```
irb(main) :002:0> load "ArrayTest.rb"
```

Output: A sample program to manipulate with Arrays in Ruby
=> true
irb(main):004:0>

```
irb(main) :003:0> my_array = ArrayTest.new(3)
```

Output: => #<ArrayTest:0x00000000a364c0 @size=3, @test_array=[nil, nil, nil]>

```
irb(main) :004:0> my_array.put_in(1,"one")
```

Output: => "one"

```
irb(main) :005:0> my_array.put_in(0,0)
```

Output: => 0

```
irb(main) :006:0> my_array.put_in(2,true)
```

Output: => true

```
irb(main) :007:0> my_array
```

Output: => #<ArrayTest:0x00000000a364c0 @size=3, @test_array=[0, "one", true]>

---

5. Introduction to RubyGems

RubyGems is a package manager for Ruby that provides a standard format for distributing Ruby programs and libraries. We might need to install some RubyGems during ECE-421 labs.

Task 4: Let us open the attached "ECE421–Using Ruby Version Manager.pdf" file, and go through it together.