University College Dublin
An Coláiste Ollscoile, Baile Átha Cliath

SEMESTER I EXAMINATION – 2014/2015

COMP 41100

Exploring Programming in Ruby

Dr. J. Westlake

Prof. P. Cunningham

Prof. M. Keane*

Time allowed: 2 hours

Instructions for candidates

Answer any FIVE Questions.
All Questions carry equal marks. Use of calculators is prohibited.

Instructions for invigilators

Use of calculators is prohibited.

1. Develop a method that tests to see if a number is a prime, as in:

   `13.is_prime?  =>  true`

   and then define two methods that use this prime-test to find the first 20 primes, checking each number counting up from 1, giving the following outputs:

   ```
   This is a prime: 2
   This is a prime: 3
   This is a prime: 5
   This is a prime: 7
   This is a prime: 11
   This is a prime: 13
   This is a prime: 17
   This is a prime: 19
   This is a prime: 23
   This is a prime: 29
   This is a prime: 31
   This is a prime: 37
   This is a prime: 41
   This is a prime: 43
   This is a prime: 47
   This is a prime: 53
   This is a prime: 59
   This is a prime: 61
   This is a prime: 67
   This is a prime: 71
   ```

   Of these two methods there should be one called (i) `find_primes1` that uses iteration and (ii) `find_primes2` that uses recursion.

   (Hint: the modulo operator in Ruby is `%`, as follows: `10 % 2 => 0`)

2. Define a class called `Tree` (with three attributes, including one called `living` which can have the value `true/false`) and a subclass of it called `ConiferousTree` (with five attributes in total, one of which is called `height`).

   Create two methods for the `Tree` superclass that are inherited by the class `ConiferousTree`, for which you should define one further method, called `classify_size`; when invoked on an instance of `ConiferousTree`, `classify_size` will return “tall” if the instance’s `height` is greater-than or equal-to 10 and “small” if the instance’s `height` is less-than 10.

   Define a module called `ClearForest` that has a method called `cut_down,` that will change the value of the `living` attribute to `false` when it is invoked on appropriate objects.

   Create a mixin, using the `ClearForest` module, such that `Tree` and `ConiferousTree` object-instances will be appropriately modified when the `cut_down` method is invoked on them.

3. Write an iterative method (using **`each, collect`** `or` **`select`**), called **`match_names`**, that takes an array of first-names (written as symbols) of any size and a list of second-names (written as strings) of any size and produces all possible pairs of the names in both lists (written as symbols with underscores). So, for example, given the two arrays:

```
[:mark, :mikki, :peijie]
["keane", "finn", "ma"]
```

the output would be:

```
[:mark_keane, :mark_finn, :mark_ma, :mikki_keane,
:mikki_finn, :mikki_ma :peijie_keane, :peijie_finn,
:peijie_ma]
```

Now, define a method – called `match_names_block` – that takes the same inputs and produces the same outputs, but does this using a block, that should be called in the following way:

`array1.match_names_block(array2){block_for_combination}`

Is it good practice to use symbols in this way? Briefly list some of the uses symbols are put to in Ruby.

4. Describe what Ruby does during *method lookup*, when an object calls a method (be it an instance or class method), how it searches for the method's definition and the conditions which lead to a `method_missing` error.

5. Write a short explanatory paragraph on all of the following topics, using appropriate examples: polymorphism, mixins, duck typing, inheritance.

6. Ruby on Rails makes use of the Model-View-Controller architecture pattern to organize the development of web-based applications. What are models, views and controllers? Write a short explanatory paragraph on each.

   Give three reasons why it might be a good idea to divide up web-based applications in this way.

7. What do the following evaluate to in Ruby:

```
i.      p "dd"
ii.     foo = "foo"; puts foo
iii.    "[a, b, c]”.instance_of?(String)
iv.     ["a","b","c].instance_of?(Array)
v.      class Egg ; end; p Egg.new
vi.     [[3], 4, 5, [6]].inject{|a, b| a + [4]}
vii.    ["a","b","c"].each{|item|  p item + "egg"}
viii.   ["a13","b22","c33"].collect{|item| item[2].to_i}
ix.     [[[2,3],[3],[4,5]]].length
x.      [1,2,3,4,4,2,3,6,2,1,145,4,3,2].uniq
xi.     float 5
xii.    "I have this thing and ".concat("another")
xiii.   ["fooble"].concat(["doodle"])
xiv.    ["fooble"] << ["doodle"]
xv.     "fooblinggg\n".chomp.chop.chop.chop
xvi.     a = 1; b = "2"; a + b
xvii.   "apples_oranges_lemons".split(/ /)
xviii.  "12345" <=> "1234"
xix.    [6,3,2,1].inject{lx,yl x / y}
xx.     a = {}; a[:foo] = "bar"; p a
```