Module 2
Pair Programming, Ruby

CS W169A: Software Engineering

1 What Would Ruby Do?

Given the following snippets of Ruby code, determine the output. If you can, find a classmate, discuss, then validate your solutions by typing the code into an interpreter. You should alternate who types and who explains the output.

(i)
```
fruit1 = "strawberry"
fruit2 = "banana"
puts fruit1.reverse
puts fruit2.reverse!
fruit1 + "␣" + fruit2
```

(ii)
```
class String
    @@hello = "hi␣there!"
    def hello; "world"; end
end
"smoothie".hello
```

(iii)
```
class Fruit
    def method_missing(meth)
        if meth.to_s =~ \/^tastes_(.+)\?\$\/
            "Yup,␣that␣fruit␣tastes␣#{\$1}!"
        else
            super
        end
    end
end
orange = Fruit.new
orange.bitter?
orange.tastes_sour?
orange.tastes_sweet?
```

2 Collections

In this next part, try to rewrite each of the following method as one (short) line. One person should be the writer,while the other person explains what to write. Try alternating roles between the two exercises. (Hint: see figure 3.7 in the textbook.)

(i)
```
def foo(arr)
    res = 0
    arr.each do |n|
        res += n
    end
    res
end
```

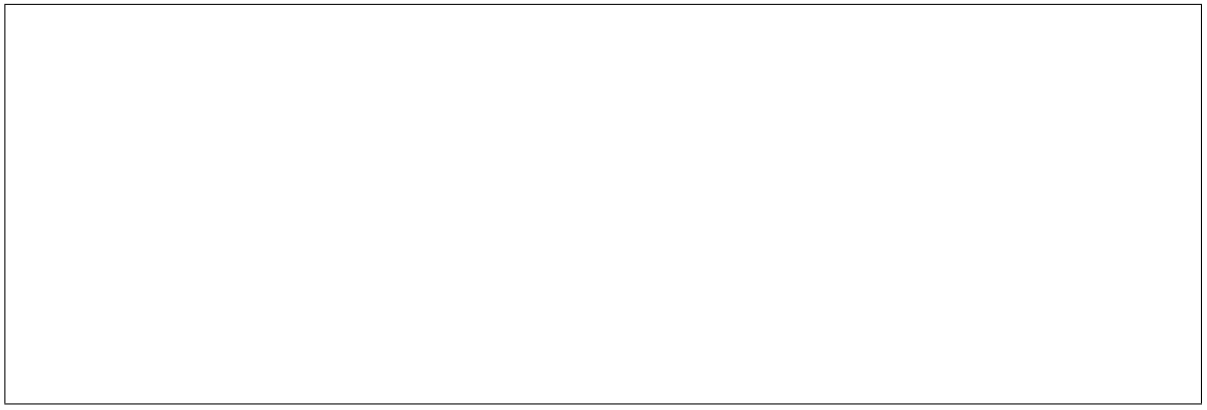
(ii)
```
def bar(hsh)
    res = {}
    hsh.each do |k, v|
        if v > 100
            res[k] = v
        end
    end
    res
end
```

3 Iterators

In this part, create your own iterators with the yield statement that return the following elements. Again, alternate roles between the two exercises.

(i) Write a function fib(n) that yields the first n Fibonacci numbers in sequence and returns nil.

```
>> fib(4) { |x| puts x }
1
1
2
3
nil
```



(ii) Write the function Array#odds which yields the odd-indexed elements of the array in sequence and returns nil.

```
>> [10, 30, 50, 70, 90].odds do |n|
..   puts n
.. end
30
70
nil
```

4 Collection Methods

Write 1 to 3 lines of Ruby code for each of the following tasks. Don't use any explicit looping (`for, loop, each, while`, etc.): use only the collection operators, most of which are defined in the module `Enumerable`. (An extra hint: the primary keyword you're going to be using is `compact`. Definitely look up the documentation online if you're not familiar with it!)

Assuming the variable words is an array in which each element is either a string (which may be empty) or `nil`, write a short amount of Ruby code that will return:

- A copy of words with nil elements removed.

- A copy of words with both nil and empty string elements removed.

- Only those words that are exactly 3 letters long.

- Only those words that contain at least one vowel (a, e, i, o, u)

- A string that is the concatenation of all the words (hint: use `inject`)

- A string that contains exactly one of each letter contained in any word, in sorted order. So if words contains `["apple", "banana", nil, "cat"]`, the string should be `"abcelnpt"`. **Hint**: Consider using `uniq`. **Hint 2**: To use `uniq`, consider also using `chars`.

5 Extra Practice

Implement a linked list. Try to include the add, delete, and contains operations.