Week 1 Section - Pair Programming in Ruby

Part One: What Would Ruby Do?

Find a partner and begin typing the following exercises into the interpreter. You should alternate who types and who explains the output.

1)
```
fruit1 = “strawberry”
fruit2 = “banana”
puts fruit1.reverse
puts fruit2.reverse!
fruit1 + “ ” + fruit2
```

2)
```
class String
  @@hello = “hi there!”
  def hello; “world”; end
end
“smoothie”.hello
```

3)
```
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?
```

Part Two: 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.)

1)
```
def foo(arr)
  res = 0
  arr.each do |n|
    res += n
  end
  res
end
```

2)
```
def bar(hsh)
  res = {}
  hsh.each do |k, v|
    if v > 100
      res[k] = v
    end
  end
  res
end
```

Part Three: Iterators

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

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
```

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
```

Extra Practice

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