CSE 341 — Ruby Discussion Questions — Solutions

1. Write a class `Delay` that implements delays (like the delay function in Scheme). The following code shows how it should work:

```
n = 0
d = Delay.new {n=n+1; 3+4}
d.force
d.force
v = d.force
e = Delay.new {1/0}
```

After we evaluate these statements `v` should be 7, but `n` should only be 1 (since we only evaluate the block once). Further, since we never force `e`, we shouldn't get a divide-by-zero error.

Solution:

```
class Delay
  def initialize(&p)
    @p = p
    @value = nil
    @unevaluated = true
  end
  def force
    if @unevaluated
      @value = @p.call
      @unevaluated = false
    end
    return @value
  end
end
```

2. Write a `min` method for the `Enumerable` mixin. You'll need to decide how to handle finding the minimum of an empty collection. Bonus points for handling this in the same way Ruby itself does!

Hint: look at the implementation of `map` at the end of the `inheritance.rb` handout.

```
def min
  m = nil
  each {|x| m = x if m.nil? or x<m}
  return m
end
```

3. Consider the class and module definitions in `self_super.rb` linked from the 341 Ruby web page. Suppose we define a class `C5` as follows:

```
class C5 < C1
  include M2
end
```

What is the result of evaluating these expressions?

```
>> x = C5.new
=> #<C5:0x35f520>
>> x.test1
in mixin M2 test1
=> nil
>> x.test2
in mixin M2 test2
in C1 test2
=> nil
>> x.kind_of?(C5)
=> true
>> x.kind_of?(M2)
=> true
>> x.kind_of?(M1)
=> false
>> C5.ancestors
=> [C5, M2, C1, Object, Kernel]
>> C5.superclass
=> C1
>> C5.superclass.superclass
=> Object
>> C5.superclass.superclass.superclass
=> nil
```