Homework 09 - Transcript-Summary-Conclusion Table & Appendix

Andrew Carter and Beryl Egerter

April 3, 2013

1 Transcript-Summary-Conclusion Table

1.1 A note on formatting

Following are annotated transcripts of ACBE4Q1, ACBE3Q1, ACBE3Q4, and ACBE1Q3. The assignment asks for these in a table format. Due to the wordy nature of the transcripts and the conclusions (and partially the summaries), we found the table format which restricts the width of each of these categories to less than a third of the page extremely cramped and almost impossible to read. In the interest of readability we have provided these in a format of:
[time]
Speaker:
transcript
Context: any context required
Summary: brief summary of corresponding transcript
Conclusions: conclusions drawn from corresponding transcript

1.2 High Level Points

The high level points we are trying to make with these transcripts are:

- (ACBE4Q1) Execution order appears to work better for the evaluation questions we have provided. Even by starting using a different method, a participant ended up executing the code instead.
- (ACBE3Q1 & ACBE3Q4)The right method can help a student move past concepts they are not familiar with. While evaluating, a student used execution order which helped him understand a concept, and switching to top down let him ignore a large portion of the code and clear up some misunderstandings.
- (ACBE1Q3) Top down order on a debugging question while not understanding many concepts used in the code can lead to someone forgetting the question.

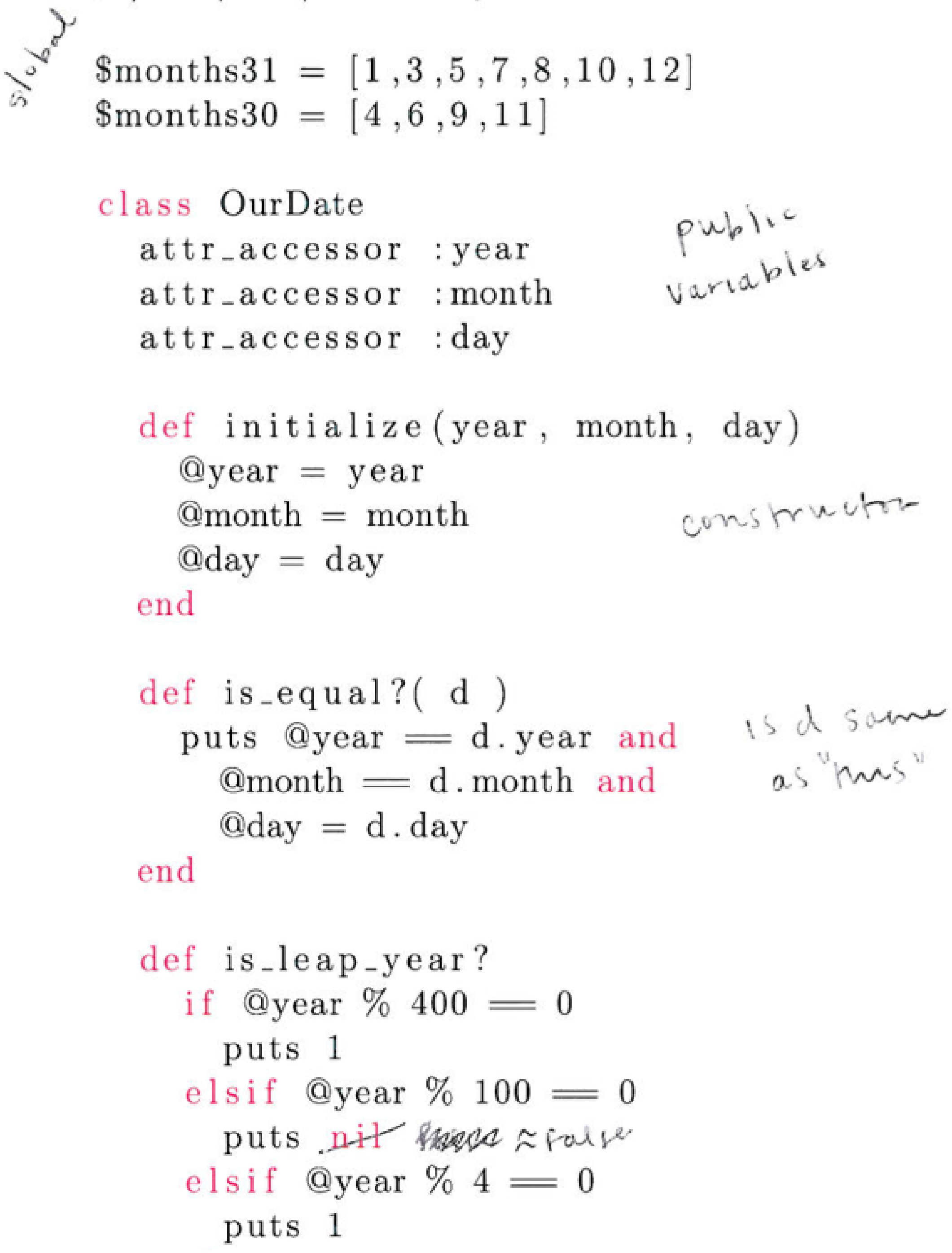
1.3 A note on context

In the context of each segment of transcript, line numbers are referred to. These line numbers correspond to line numbers of code within the Appendix (Section 2 of this document). Also used are figures which are scanned copies of the paper that the students wrote on during the interviews. The figures follow the annotated transcripts as Section 1.5 of this document.

1.4 remove this page and put in google doc pages here instead

1.5 Figures

Fig 1. ACBE1Q3



```
global
$months31 = [1,3,5,7,8,10,12]
$months30 = [4,6,9,11]

class OurDate
  attr_accessor :year              public
  attr_accessor :month             variables
  attr_accessor :day

  def initialize(year, month, day)
    @year = year
    @month = month                 constructor
    @day = day
  end

  def is_equal?( d )
    puts @year == d.year and       is d same
      @month == d.month and        as "this"
      @day = d.day
  end

  def is_leap_year?
    if @year % 400 == 0
      puts 1
    elsif @year % 100 == 0
      puts ~~nil~~ [illegible] ≈ false
    elsif @year % 4 == 0
      puts 1
```

Fig 2. ACBE4Q1

Question 1.1

```
def func2(list , num):
    return func1(list , num, func4)

def func4(a, b):
    return a * b

def func1(list , num, f):
    acc = 0
    for i in list:
        acc += f(i , num)
    return acc

def main():
  print(func3([1 ,2 ,3 ,4]))

def func3(list):
  return func2(list , 4)

main()
```

*

fun3([1, 2, 3, 4])

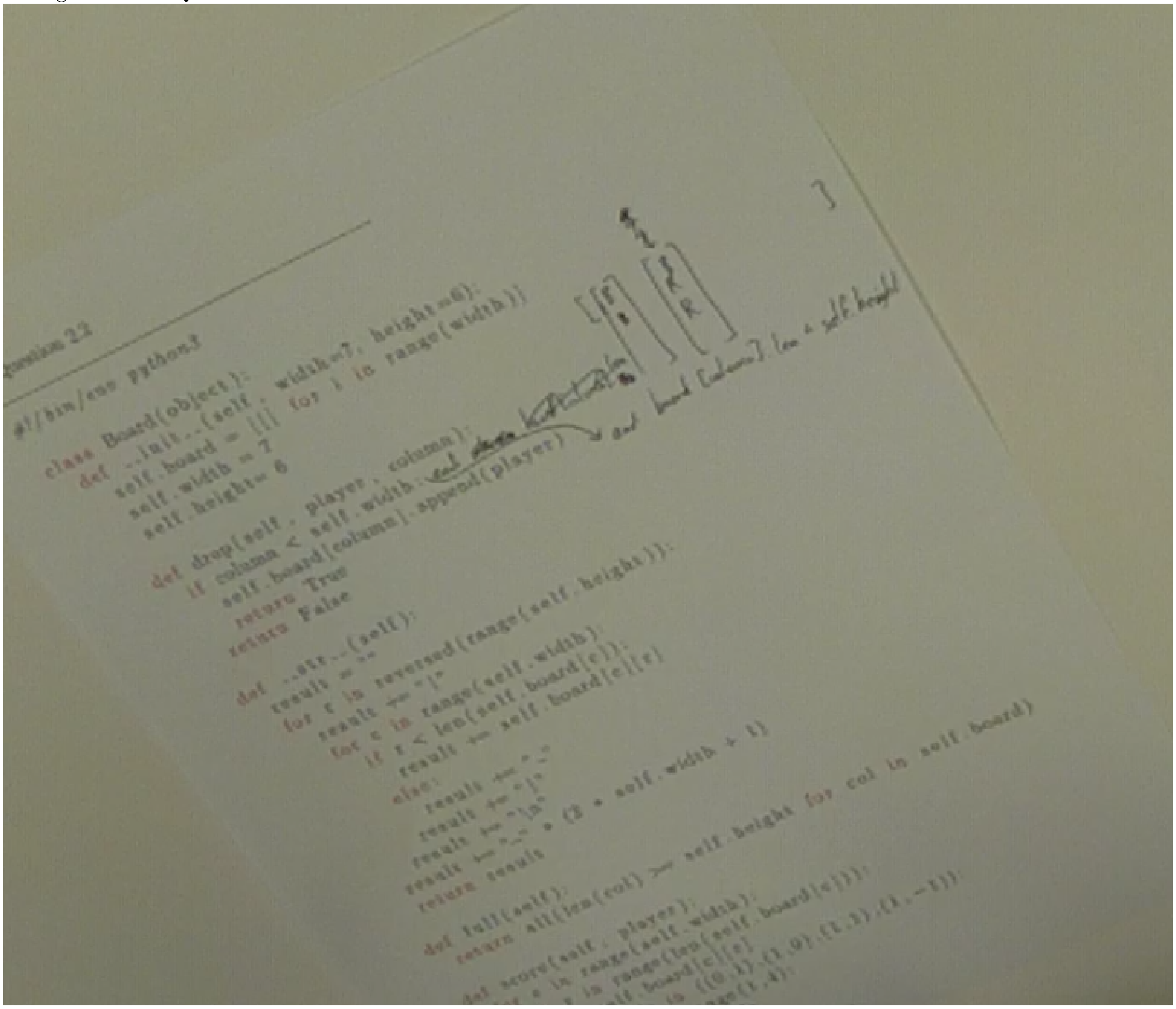
func2([1, 2, 3, 4], 4)

func1([1, 2, 3, 4], 4, *)

acc = 0		0
+=	1 * 4	4
+=	2 * 4	12
+=	3 * 4	24
+=	4 * 4	40

"40"

Fig 3. ACBE3Q4



question 22

```
#!/bin/env python3

class Board(object):
    def __init__(self, width=7, height=6):
        self.board = [[] for i in range(width)]
        self.width = 7
        self.height= 6

    def drop(self, player, column):
        if column < self.width:
            self.board[column].append(player)
            return True
        return False

    def __str__(self):
        result = ""
        for r in reversed(range(self.height)):
            result += "|"
            for c in range(self.width):
                if r < len(self.board[c]):
                    result += self.board[c][r]
                else:
                    result += "_"
                result += "|"
            result += "\n"
        result += "_" * (2 * self.width + 1)
        return result

    def full(self):
        return all(len(col) >= self.height for col in self.board)

    def score(self, player):
        for c in range(self.width):
            for r in range(len(self.board[c])):
                    elf.board[c][r]
                    in ((0,1),(1,0),(1,1),(1,-1)):
                    nge(1,4):
```

and board[column].len < self.height

2 Appendix

2.1 Interview Question 1

Code given to participants did not include line numbers.
Code given to ACBE1, ACBE2, and ACBE3 was in color. Code given to ACBE4 was in grayscale.
Verbal prompt was given before handing code to participant:
"For this question we would like you to familiarize yourself with some Python code. Please explain to us what you think this code does."

```python
def func2(list, num):
  return func1(list, num, func4)

def func4(a, b):
  return a * b

def func1(list, num, f):
  acc = 0
  for i in list:
      acc += f(i, num)
  return acc

def main():
  print(func3([1,2,3,4]))

def func3(list):
  return func2(list, 4)

main()
```

2.2 Interview Question 2

Code given to participants did not include line numbers.
Code given to ACBE1, ACBE2, and ACBE3 was in color. Code given to ACBE4 was in grayscale.
Verbal prompt was given before handing code to participant:
"For this question, we would like you to again familiarize yourself with some Python code. Please explain to us what you think this code does. "

```python
def function50(i, L):
  return L[i+2]

def function37(L):
  return [L[-1]]+L

def function52(i):
  return function4() * i

def function1(j, k):
  return (j + k) * function52(1)

def function4():
  return 3

def function188(L):
  return function37(L)+[function50(2, L)]

def function0():
  return function188([1,2,3,4,5,6,7,8,9])[function1(0,1)]

x = function0()
print x
```

2.3 Interview Question 3

Code given to participants did not include line numbers.
Code given to ACBE1, ACBE2, and ACBE3 was in color. Code given to ACBE4 was in grayscale. Code given to ACBE1 and ACBE2 had double equals signs that were joined together. Code given to ACBE3 and ACBE4 had spaces between the equals signs.
Verbal prompt was given before handing code to participant:
"For this question we would like to have you look at some code in the programming language Ruby. This is the scenario: A coworker recently left on vacation and left two files behind. One of those files is a Date class and your boss wasn't sure if the coworker had finished including leap year support. Your boss would like you to make sure it is supported."

File 1:

```
#!/usr/bin/ruby

load "ourdate.rb"

d = OurDate.new(2011,1,4)
print "#{d.what_day}"
print "We started writing this file today.\n"
d.forward_time(365)
print "We are almost done now.\n"
print "#{d.what_day}"
```

File 2

```
#!/usr/bin/env ruby

$months31 = [1,3,5,7,8,10,12]
$months30 = [4,6,9,11]

class OurDate
  attr_accessor :year
  attr_accessor :month
  attr_accessor :day

  def initialize(year, month, day)
    @year = year
    @month = month
    @day = day
  end

  def is_equal?( d )
    puts @year = = d.year and
      @month = = d.month and
      @day = d.day
  end

  def is_leap_year?
    if @year % 400 = = 0
      puts true
    elsif @year % 100 = = 0
      puts false
    elsif @year % 4 = = 0
      puts true
    else
      puts false
    end
  end

  def check_month
    if @month == 13
      @month = 1
      @year = @year + 1
    elsif @month == 0
      @month = 12
      @year = @year - 1
    end
  end

  def tomorrow
    @day = @day + 1
    if @day > 31
      for i in $months31
        if @month == i
          @day = 1
          @month = @month + 1
          check_month
        end
      end
    elsif @day > 30
      for i in $months30
        if @month == i
          @day = 1
          @month = @month + 1
          check_month
        end
      end
    elsif @day > 28 and @month == 2
      @day = 1
      @month = @month + 1
      check_month
    end
  end

  def yesterday
    @day = @day - 1
    if @day == 0
      @month = @month - 1
      check_month
      for i in $months31
        if @month == i
          @day = 31
        end
      end
      for i in $months30
        if @month == i
          @day = 30
        end
      end
      if @month == 2
        @day = 28
      end
    end
  end

  def forward_time(n)
    for i in 0..n
      tomorrow
    end
  end

  def reverse_time(n)
    for i in 0..n
      yesterday
    end
  end

  def what_day
    puts "Today is #{month}/#{day}, #{year}!"
  end
end
```

2.4 Interview Question 4

Code given to participant did not contain line numbers.
Code given to ACBE1, ACBE2, and ACBE3 was in color. Code given to ACBE4 was in grayscale. Code given to ACBE1 and ACBE2 had double equals signs that were joined together. Code given to ACBE3 and ACBE4 had spaces between the equals signs.
Verbal prompt was given before handing code to participant:
"For this question we would like to have you look at some code in Python. This is the scenario: You acquired a connect 4 program from a friend. However, the friend has warned you that you can put too many pieces in a column. Determine a possible fix for this bug so that you can enjoy your connect 4 program."

```python
#!/bin/env python3

class Board(object):
  def __init__(self, width=7, height=6):
    self.board = [[] for i in range(width)]
    self.width = 7
    self.height= 6

  def drop(self, player, column):
    if column < len(self.board):
      self.board[column].append(player)
      return True
    return False

  def __str__(self):
    result = ""
    for r in reversed(range(self.height)):
      result += "|"
      for c in range(self.width):
        if r < len(self.board[c]):
          result += self.board[c][r]
        else:
          result += " "
        result += "|"
      result += "\n"
    result += "-" * (2 * self.width + 1)
    return result

  def full(self):
    return all(len(col) >= self.height for col in self.board)

  def score(self, player):
    for c in range(self.height):
      for r in range(len(self.board[c])):
        p = self.board[c][r]
        for dc,dr in ((0,1),(1,0),(1,1),(1,-1)):
          for i in range(1,4):
            nc = c + i*dc
            nr = c + i*dr
            if nc < 0 or self.width <= nc:
              break
            if nr < 0 or len(self.board[nc]) <= nr:
              break
            if self.board[nc][nr] != p:
              break
          else:
            return 1 if p = = player else -1
    return 0

```

```
other = {'X' : 'O', 'O' : 'X'}
player = 'X'
board = Board()

while True:
  try:
    c = int(input("%s > " % player))
  except TypeError:
    continue
  if not board.drop(player,c):
    continue
  print(board)
  if board.score(player):
    print("Player %s Wins!!!" % player)
  elif board.full():
    print("Tie")
  else:
    player = other[player]
    continue
  board = Board()
  player = 'X'
  print(board)
```