COMP10001 Foundations of Computing
Practice Exam

Semester 2, 2015

Chris Leckie

September 16, 2015

Question 2

Consider the following code fragment:

```
x = [("andrew",True), ("kim",False),
     ("sandy",True), ("tim",True)]
limit = 2
i = 0
j = len(x) - 1
y = []
while i < limit:
    if x[j][1]:
        y.append(x[j][0])
        i += 1
    j -= 1
```

List the values of the variables `i` and `j` after each iteration of the loop.

Question 2 (cont.)

iteration	`i`	`j`
1	1	2
2	2	1

Question 2 (cont.)

Consider the following code fragment:

```
x = [("andrew",True), ("kim",False),
     ("sandy",True), ("tim",True)]
limit = 2
i = 0
j = len(x) - 1
y = []
while i < limit:
    if x[j][1]:
        y.append(x[j][0])
        i += 1
    j -= 1
```

What is the final value of `y`?

Question 2 (cont.)

Consider the following code fragment:

```
x = [("andrew",True), ("kim",False),
     ("sandy",True), ("tim",True)]
limit = 2
i = 0
j = len(x) - 1
y = []
while i < limit:
    if x[j][1]:
        y.append(x[j][0])
        i += 1
    j -= 1
```

What is the final value of `y`?

A: *`['tim', 'sandy]`*

Question 3

The following program is meant to determine whether a string contains only alphabetic characters, but the lines are out of order. Put the line numbers in the correct order and introduce appropriate indentation (indent the line numbers to show how the corresponding lines would be indented in your code).

```
1     assert(all_letters("abCD"))
2     if not 'A' <= letter <= 'Z':
3     return False
4     for letter in string:
5     return True
6     def all_letters(string):
7     if not 'a' <= letter <= 'z':
8     assert(not all_letters("abCD123"))
```

Question 3 (cont.)

```
6
    4
        2
            7
                3
    5
1
8
```

Question 4

```
1  def get_message(filename, which):
2      text = open("filename").read()
3      lines = text.split(',')
4      message = ''
5      for line in lines:
6          if which == 1:
7              index = len(line)
8          else:
9              index = 0
10         message += line(index)
11     return message
```

Identify exactly three (3) errors and specify: (a) the line number where the error occurs; (b) the type of error; and (c) how you would fix each error.

Question 4 (cont.)

A:

- line 2 (run-time [if no file of name "filename"] OR logic): `text = open(filename).read()`
- line 3 (logic): `lines = text.split('\n')`
- line 7 (run-time [on line 10]):

  `index = len(line) - 1`
- line 10 (run-time): `message += line[index]`
- line 10 (logic): `message += line[index].upper()`

Question 5

```
import [ 1 ]
data = open("sales.csv")
sales = list(csv.DictReader(data))
totalsales = [ 2 ]
for row in sales:
    salesman = row['Salesman']
    profit = [ 3 ]
    if salesman [ 4 ] totalsales:
        totalsales[salesman] = 0
    [ 5 ] += profit
max_salesman = ''
max_profit = [ 6 ]
for salesman in totalsales:
    if max_profit [ 7 ] totalsales[salesman]:
        max_profit = totalsales[salesman]
        max_salesman = [ 8 ]
print("{0} earns the highest profit of {1}".format(max_salesman, ma
```

Question 5 (cont.)

```
import [ csv ]
data = open("sales.csv")
sales = list(csv.DictReader(data))
totalsales = [ 2 ]
for row in sales:
    salesman = row['Salesman']
    profit = [ 3 ]
    if salesman [ 4 ] totalsales:
        totalsales[salesman] = 0
    [ 5 ] += profit
max_salesman = ''
max_profit = [ 6 ]
for salesman in totalsales:
    if max_profit [ 7 ] totalsales[salesman]:
        max_profit = totalsales[salesman]
        max_salesman = [ 8 ]
print("{0} earns the highest profit of {1}".format(max_salesman, ma
```

Question 5 (cont.)

```
import [      csv      ]
data = open("sales.csv")
sales = list(csv.DictReader(data))
totalsales = [      {}      ]
for row in sales:
    salesman = row['Salesman']
    profit = [      3      ]
    if salesman [      4      ] totalsales:
        totalsales[salesman] = 0
    [      5      ] += profit
max_salesman = ''
max_profit = [      6      ]
for salesman in totalsales:
    if max_profit [      7      ] totalsales[salesman]:
        max_profit = totalsales[salesman]
        max_salesman = [      8      ]
print("{0} earns the highest profit of {1}".format(max_salesman, ma
```

Question 5 (cont.)

```
import [        csv        ]
data = open("sales.csv")
sales = list(csv.DictReader(data))
totalsales = [        {}         ]
for row in sales:
    salesman = row['Salesman']
    profit = [ int(row['Profit']) ]
    if salesman [        4        ] totalsales:
        totalsales[salesman] = 0
    [        5        ] += profit
max_salesman = ''
max_profit = [        6        ]
for salesman in totalsales:
    if max_profit [        7        ] totalsales[salesman]:
        max_profit = totalsales[salesman]
        max_salesman = [        8        ]
print("{0} earns the highest profit of {1}".format(max_salesman, ma
```

Question 5 (cont.)

```
import [        csv        ]
data = open("sales.csv")
sales = list(csv.DictReader(data))
totalsales = [        {}        ]
for row in sales:
    salesman = row['Salesman']
    profit = [ int(row['Profit']) ]
    if salesman [      not in      ] totalsales:
        totalsales[salesman] = 0
    [        5        ] += profit
max_salesman = ''
max_profit = [        6        ]
for salesman in totalsales:
    if max_profit [        7        ] totalsales[salesman]:
        max_profit = totalsales[salesman]
        max_salesman = [        8        ]
print("{0} earns the highest profit of {1}".format(max_salesman, ma
```

Question 5 (cont.)

```
import [ csv ]
data = open("sales.csv")
sales = list(csv.DictReader(data))
totalsales = [ {} ]
for row in sales:
    salesman = row['Salesman']
    profit = [ int(row['Profit']) ]
    if salesman [ not in ] totalsales:
        totalsales[salesman] = 0
    [ totalsales[salesman] ] += profit
max_salesman = ''
max_profit = [ 6 ]
for salesman in totalsales:
    if max_profit [ 7 ] totalsales[salesman]:
        max_profit = totalsales[salesman]
        max_salesman = [ 8 ]
print("{0} earns the highest profit of {1}".format(max_salesman, ma
```

Question 5 (cont.)

```
import [ csv ]
data = open("sales.csv")
sales = list(csv.DictReader(data))
totalsales = [ {} ]
for row in sales:
    salesman = row['Salesman']
    profit = [ int(row['Profit']) ]
    if salesman [ not in ] totalsales:
        totalsales[salesman] = 0
    [ totalsales[salesman] ] += profit
max_salesman = ''
max_profit = [ -1 ]
for salesman in totalsales:
    if max_profit [ 7 ] totalsales[salesman]:
        max_profit = totalsales[salesman]
        max_salesman = [ 8 ]
print("{0} earns the highest profit of {1}".format(max_salesman, ma
```

Question 5 (cont.)

```
import [ csv ]
data = open("sales.csv")
sales = list(csv.DictReader(data))
totalsales = [ {} ]
for row in sales:
    salesman = row['Salesman']
    profit = [ int(row['Profit']) ]
    if salesman [ not in ] totalsales:
        totalsales[salesman] = 0
    [ totalsales[salesman] ] += profit
max_salesman = ''
max_profit = [ -1 ]
for salesman in totalsales:
    if max_profit [ < ] totalsales[salesman]:
        max_profit = totalsales[salesman]
        max_salesman = [ 8 ]
print("{0} earns the highest profit of {1}".format(max_salesman, ma
```

Question 5 (cont.)

```
import [ csv ]
data = open("sales.csv")
sales = list(csv.DictReader(data))
totalsales = [ {} ]
for row in sales:
    salesman = row['Salesman']
    profit = [ int(row['Profit']) ]
    if salesman [ not in ] totalsales:
        totalsales[salesman] = 0
    [ totalsales[salesman] ] += profit
max_salesman = ''
max_profit = [ -1 ]
for salesman in totalsales:
    if max_profit [ < ] totalsales[salesman]:
        max_profit = totalsales[salesman]
        max_salesman = [ salesman ]
print("{0} earns the highest profit of {1}".format(max_salesman, ma
```

Question 6

Complete a Python program that produces an HTML table containing the taxicab distances between any pair of cities. The table should be similar to the following figure.

	A	B	C	D	E
A	0.0	2.0	4.0	4.0	8.0
B	2.0	0.0	4.0	4.0	8.0
C	4.0	4.0	0.0	4.0	4.0
D	4.0	4.0	4.0	0.0	4.0
E	8.0	8.0	4.0	4.0	0.0

Question 6 (cont.)

The program has been started for you; you need to write the code that generates the distance table.

```
import csv

def taxicab_dist(x1,y1,x2,y2):
    return abs(x2-x1) + abs(y2-y1)

data = csv.reader(open("cities.csv"))
header = data.next()
rows = list(data)
num_cities = len(rows)

print('<!DOCTYPE html><html><body>')
# write your code from here
```

Question 6 (cont.)

```
cities = {}
for row in rows:
    cities[row[0]] = (float(row[1]),float(row[2]))
print("<table border='1'><tr><td></td>")
for city in sorted(cities):
    print("<td>{0}</td>".format(city))
print("</tr>")
for city in sorted(cities):
    print("<tr><td>{0}</td>".format(city))
    for city2 in sorted(cities):
        print("<td>{0}</td>".format(\
            taxicab_dist(cities[city][0],\
            cities[city][1],cities[city2][0],\
            cities[city2][1])))
    print("</tr>")
print("</table>")
print("</body></html>")
```

Question 7a

What is an "algorithm" in the context of Computing?

Question 7a

What is an "algorithm" in the context of Computing?

A: *An algorithm is a set of steps for solving an instance of a particular problem type*

Question 7b

Outline the "generate-and-test" strategy of algorithmic problem solving. With the aid of an example, explain what sort of problems it is commonly applied to.

Question 7b

Outline the "generate-and-test" strategy of algorithmic problem solving. With the aid of an example, explain what sort of problems it is commonly applied to.

A: *Generate candidate answers and test them one by one until a solution is found; assumes a candidate answer is easy to test and that the set of candidate answers is ordered or can be generated exhaustively*

Question 8

Consider a blank USB key with a capacity of 4 gigabytes (assume that 1 gigabyte = 1,000,000,000 bytes). A wedding photographer intends to use this 4GB key to distribute all of their works. The photographer takes each of the digital images as a RGB colour image *4000 pixels wide* and *3000 pixels high*. To save space, these images are compressed using JPEG compression. This will reduce the space required to store the images to *one-tenth* (1/10) of the original space.

Calculate the maximum number of images that can be stored on the thumbdrive, using JPEG compression.

Question 8 (cont.)

A:

$$\begin{aligned} \text{images} &= \left\lfloor \frac{4000000000}{4000 \times 3000 \times 3 \times 0.1} \right\rfloor \\ &= 1111 \end{aligned}$$