CS116 Final Exam Review!

July 27\textsuperscript{th}, 2018 – Doomsday is Near
Best Way to Prepare

- Review Assignment feedback (MarkUs) and solutions (Learn)
- Review your notes
- Review Tutorial Questions and solutions
- Practice Handwriting Code (example questions on course webpage > Exams
- Practice tracing code
- Seek assistance as needed (We are here to help!)
- Sleep!
Pre-Review Q&A

- Old Assignment Questions?
- Course Questions?
ISA Exam Review Questions
Question 1: Shipping Fiesta

When shipping a parcel, there are 3 main factors to consider: weight, destination, and speed of delivery. The following rules apply:

- There is a $5 flat rate fee.
- If $0 < weight \leq 100$, then the charges per gram are $0.05
- If weight > 100, then the charges per gram are $0.08
- If destination is Canada, then no additional charges are added.
- If destination is anything other than Canada, add $15
- If speed is 2 (i.e: 2-day delivery), add $17.99.
- If speed is 3 (i.e: 3-day delivery), add $12.99. Otherwise, add $6.99. Finally, add a 15% tax to the total.

Write a Python function called `charges` which consumes 3 parameters, `weight` (in grams), `destination` (a string), and `speed` (an integer), and produces the total charges to ship the package based on the information above.
Question 2: Box_Ception

Write a Python function called `box` which consumes a single character, `s`, and a natural odd number, named `n`, and prints to the screen a box shape made of “s”. It will consist of “n” number of lines.

Eg.

```
box("X",5) => None and prints:
xxxxxx
X   X
X   X
X   X
X   X
xxxxxx
```
Question 3: L33T Speak

L33T speak is a different form of writing, usually online. It consists of replacing certain letters with symbols or numbers. Consider the table below of letters, and their corresponding L33T translations.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>K</th>
<th>O</th>
<th>S</th>
<th>T</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>@</td>
<td>8</td>
<td>(</td>
<td>)</td>
<td>3</td>
<td>9</td>
<td>#</td>
<td>1</td>
<td>&lt;</td>
<td>0</td>
<td>$</td>
<td>+</td>
<td>%</td>
</tr>
</tbody>
</table>

Write a Python function called `leet` which consumes a list of strings, `los`, and mutates the list where each string becomes its leet version. The function produces `None`.

For example, if `L = ["password", "secrets", “CS116”]`, then we call `leet(L)`, the list is mutated to become `[‘p@$$w0r’, ‘$3(r3+’, ‘{$116’]`. 
Question 4: How Efficient Am I?

Determine the Runtime of the Following:

```python
def f(L):
    if L == []:
        return True
    elif L[0] > L[1]:
        return f(L[1:])
    else:
        return f(L[2:])

def g(L):
    if L == []:
        return True
    elif L[0] > 0 and f(L[1:]):
        return False
    else:
        return f(L[2:])
```
Question 5: Palindrome

Write a Python function called `palindrome` which consumes a string `s`, and returns `True` if the string is a palindrome, `False` otherwise.

(a palindrome is a word that is spelt the same backwards and forwards)

Eg.

`palindrome("racecar")` => True
`palindrome("Racecar")` => False
`palindrome("John will give us all 100% on the Exam")` => False

Provide Structural, Accumulative and Generative recursive solutions.
Write a Python function called `sweeper` which consumes a list of strings `los`, and a string `s`. The function **mutates** `los`, such that all occurrences of `s` in `los` are **removed**. The function returns a list of naturals representing the number of `s`’s removed in each string within `los`.

Eg.

My_lst = ["abc", "cs116W18", "10 is the best number"]

`sweeper(My_lst, "1")` => `[0, 3, 1]` and `My_lst` is mutated to:

["abc", "cs6W8", "0 is the best number"fsp]
Question 7 – 9: Domo Arigato Mr. Roboto

"Who’s RA9!?"
Robot Class

Class Robot:

'''Fields:
direction: A string (Any of N, E, S, W)
move: A list of Str
battery: A natural
model: A string'''

def __init__(self, direction, move, battery, model):
    self.direction = direction
    self.move = move
    self.battery = battery
    self.model = model
def __repr__(self):
    return "{0} is facing {1}, and has {2} battery left.\nRobot has moved: {3}".format(self.model, self.direction, self.battery, self.move)

def __eq__(self, other):
    return isinstance(other, Robot) and \
    other.direction == self.direction and \
    other.move == self.move and \
    other.battery == self.battery and \
    other.model == self.model
Question 7: Order the Bots

Write a Python function called `create_dict` that consumes a list of Robots, `loR`, and produces a dictionary. The keys of the dictionary are the `model` numbers of the Robot. The values are a list of the Robot classes with the same model.

Eg.

```python
Robo1 = Robot("N", [], 100, "RA9")
Robo2 = Robot("W", [], 99, "RA9")
Robo3 = Robot("E", [], 10, "RA8")
create_dict([Robo1, Robo2, Robo3]) =>
{"RA9" : [Robot("N", [], 100, "RA9"), Robot("W", [], 99, "RA9")],
 "RA8" : [Robot("E", [], 10, "RA8")]}
Write a Python function called `rotate` which consumes a Robot `r` and an integer `degree` and returns a Robot in which it’s direction has been changed.

Degrees move in a counter-clockwise direction and `degree` must be a value divisible by 90.

The Robot’s direction is changed by the degree. For each multiple of 90 in `degree`, one point of battery must be deducted from the Robot. If the battery falls to 0 before finished rotating, the function will instead return “Not Enough Battery!”

Eg.

- `rotate(Robot("N", [], 100, "RA9"), 90) => Robot("W", [], 100, "RA9")`
- `rotate(Robot("N", [], 100, "RA9"), 180) => Robot("S", [], 100, "RA9")`
- `rotate(Robot("N", [], 100, "RA9"), 360) => Robot("N", [], 100, "RA9")`
- `rotate(Robot("N", [], 2, "RA9"), 360) => “Not Enough Battery!”`
**Methods are internal class functions***

Write a Python method called `direct` which consumes a Robot `self`. The function prompts for user input, with the following rules:
1. Upon start up, the robot’s move field must be the empty list.

2. The function prompts the user with “Where should <robot model> go?: “ where <robot model> is formatted into the Robot Class’s model.

3. The function will only accept the input: N, W, E, S, back, stop and distance. If the user provides invalid input, the function will print, “Invalid Direction” and redisplay the prompt above.

4. The Robot’s move and direction field will be updated, where move will append the new value if it is one of N, E, S, W and the direction of the Robot self will be updated.

5. If the user input is “back”, then the last item in move must be popped, and direction must be reversed (N to S and E to W). If the move field is empty the function will print “Cannot Reverse” and prompt for input.

6. 1 battery must be deducted per successful move (forward and backwards). If battery falls to 0, the function will print “All Out of Battery” and return the Robot.

7. If the user input’s “distance”, the function will print all the elements in move in order, each item joined with a “->”.

8. If the user inputs “stop”, the function will print “<robot model> is done moving!” where <robot model> is formatted into the robot’s model and then print all the movements the robot performed with the same format as step f. The last element should be “stop” in the chain of movements. The function will then return r.
Robo = Robot("N", [], 100, "RA9")
Robo.direct()
>>"Where should RA9 go?: N"
>>"Where should RA9 go?: South"
Invalid Input
>>"Where should RA9 go?: S"
>>"Where should RA9 go?: W"
>>"Where should RA9 go?: back"
>>"Where should RA9 go?: E"
>>"Where should RA9 go?: distance"
N->S->E
>>"Where should RA9 go?: N"
>>"Where should RA9 go?: N"
>>"Where should RA9 go?: N"
>>"Where should RA9 go?: stop"
N->S->E->N->N->stop
The Caesar Cipher is a type substitution cipher in which each letter in the alphabet are shifted down the alphabet by a value. For instance, if the shift value is 1 then all A’s will be B’s, all B’s will be C’s, all C’s will be D’s, etc...

Write a Python function called `ceasar_cipher` which consumes a `filename` and a natural `shift`. The function then opens `filename` for reading, shifting all alphabetical characters by the shift value in the alphabet. The result is written to a file called “filename_cipher.txt”. The function returns None.

Eg.
If shift = 3 and filename = “S18Exam.txt” with the contents:
Question 1: “What is the Definition of a Basis?”

The function will write to a file named “S18Exam_cipher.txt”:
txhvwrq 1: “zkdw lv wkh ghiqlwlrq ri d edvlv?”

Note that capitalization does not matter, the resulting text is all lowercase. Non-alphabetical characters are ignored in translation.
Write a Python function, called `breadth_first`, which consumes a graph, $G$, represented by an adjacency list, and a vertex, $v$. The function should use breadth-first traversal, starting at $v$, and produce the list (queue) of all vertices visited in the proper order.
Question 12: Feeling Dense

Write a Python function called `density` which consumes a graph, \( G \), and returns the density of the graph. The density is calculated as:

\[
D = \frac{(2 \cdot E)}{[V \cdot (V - 1)]}
\]

Where \( E \) is the number of edges in the graph, and \( V \) is the number of vertices. Assume no edges loop back to the vertex it is coming from. Assume \( V > 1 \).
Final Thoughts

- Don’t be Nervous, An Exam is an Exam and a grade will not define you
- Study hard, practice often, and make sure to get a good night’s sleep before the exam
- Know your exam room and seat in advance!
- Get that easy

💯 💯 💯
"It's finally over"
Thank You For an Awesome Term!

Lazeez for Everyone! (JK, I’m poor)