CSE101: End-term Review

These are some questions to help you review for the final exam, with answers. This is designed to help with your studying, but **DO NOT** assume that I will only ask questions like the ones present on this review. The final exam will be structured similar to the quizzes and since you have access to your computer, the questions will not likely ask you the output of running code.

1. A. Match the following pairs.

Computation	A description of solving the problem which includes a precise statement of the problem (the input), the desired solution (the output), and the order in which the steps will be executed
Algorithm	A problem might not be solvable by computation because it is ambiguous, it requires too many steps to complete, or it is mathematically impossible
Limitations	sequence of simple, well-defined steps carried out to solve a problem

B. Give an example of problem which is computationally huge such that it is unsolvable by a computer.

2. A. Given x = 10, what will be the result of evaluating the following expressions?

```
((x / 2) ** 2) / 2 =
(x / 2) ** 2 / 2 =
x / 2 ** 2 / 2 =
```

B. Suppose the following strings are defined in an interactive session:

```
>>> s = "baseball"
>>> t = "The Korea Baseball Championship, is the highest level league
of baseball in South Korea."
```

What will Python print as the value of the following expressions?

```
(s + '!') * 2 =
s in t =
t.count('o') =
```

C. Define a function named `pmt` that will compute and return the amount of a monthly payment on a loan. The three parameters of the function should be amt, the initial loan amount; rate, the annual interest rate and yrs, the number of years before the loan is paid off. The algorithm for computing the payment is as follows. First, calculate a value r using the formula `r = rate/100/12`. Then calculate a value `p = 12 x yrs`. The formula for the payment is then:

$$\texttt{payment} = \frac{r \times \texttt{amt}}{1 - (1 + r)^p}$$

3. A. Match the following pairs:

prime number	A function that implements a complete solution to a problem
composite number	A function designed to solve a small part of a larger problem
top level function	An integer that is not evenly divisible by any numbers except 1 and itself
helper function	An integer that can be expressed as the product to two or more other integers

B. Suppose a list is defined with this statement:

```
>>> gas = ['He', 'Ne', 'Ar', 'Kr', 'Xe', 'Rn']
```

What are the values of the following Python expressions?

```
'Ne' in gas =
'Fe' in gas =
gas.index('Ne') =
gas.index('Xe') =
```

C. Suppose we define two lists of numbers as follows:

```
>>> a = [1,1,2,3,5,8]
>>> b = [13,21,34]
```

Explain what are the values of the following Python expressions?

```
a[0] + b[0] =
a + b =
```

4. A. Assume a list is defined with this statement:

```
>>> heavy = ['U', 'Np', 'Pu', 'Am', 'Cm', 'Bk', 'Cf']
```

Show how the list will look after each step when `isort` (insertion sort) sorts a value in the array. The first line is given to get you started:

```
>>> isort(heavy)
['U'] ['Np', 'Pu', 'Am', 'Cm', 'Bk', 'Cf']
```

B. Define a Python function named `gcd` that will compute the greatest common divisor of two integers a and b using Euclid's algorithm. The pseudocode of this algorithm is given below. In modern terminology, the algorithm uses a while loop that terminates when a = b. In the body of the loop, compare a to b and subtract the smaller value from the larger one. When the loop terminates, return the result of the call.

Pseudocode of Euclid's GCD algorithm:
(:= represents assignment)

```
function gcd(a, b)
    while a ≠ b
        if a > b
           a := a - b;
        else
           b := b - a;
    return a;
```

Corresponding Python code:

5. A. Match the following pairs:

divide and conquer	An algorithm that sorts a list through a top-down application of the divide an conquer strategy
merge sort	A problem that can be broken into one or more subproblems that are each smaller instances of the main problem
quick sort	A problem-solving strategy that breaks a problem into smaller pieces and addresses each subproblem separately
recursive problem	An algorithm that sorts a list by combining small groups into larger groups, using a bottom-up application of the divide and conquer strategy

B. Write both, an iterative function and a recursive function that returns the n-th number in a Fibonacci series 0, 1, 1, 2, 3, 5, 8, 13, 21, …..

6. A. Write an assignment statement that create dictionary named continents for the seven continents on the earth which are: `Asia, Africa, North America, South America, Europe, Oceania, Antarctica`. In this dictionary, the first two characters of the continent name is the key and the name of the continent is the value.

   B. Write a function, *acronym*, that creates an acronym from the first letter of each long word in a list, where a long word is any word with more than three letters.

```
>>> acronym('operating system')
'OS'
>>> acronym('association for computing machinery')
'ACM'
```

7. (Pseudo-Random Numbers) Study the random number generator code here:

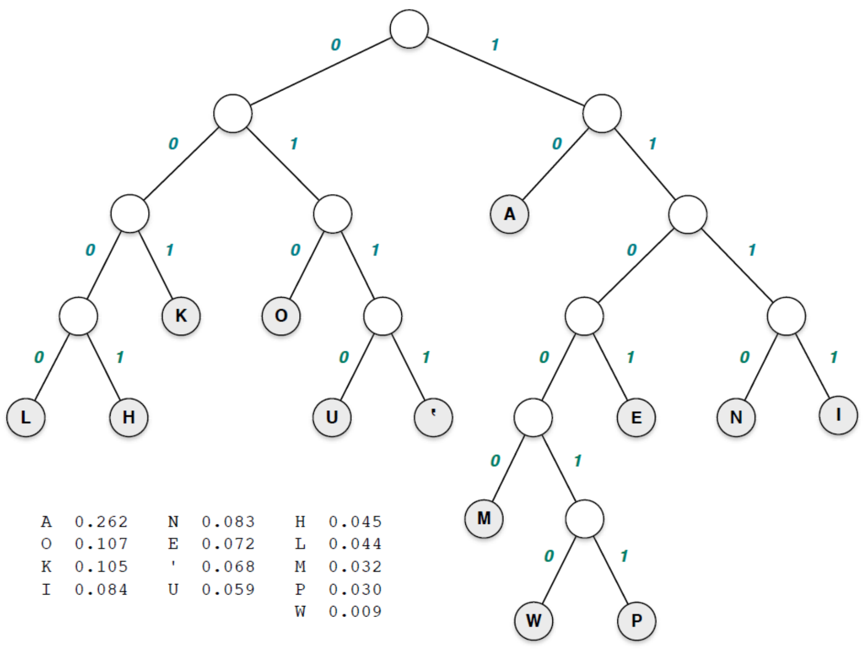
```
a = 4
c = 11
m = 23
x = 3 % m

def rand():
    global x
    x = (a * x + c) % m
    return x

for i in range(10):
    print (str(rand()) + " ", end="")
```

What 10 values will this code generate?

8. (Huffman Codes)



A	0.262	N	0.083	H	0.045		
O	0.107	E	0.072	L	0.044		
K	0.105	'	0.068	M	0.032		
I	0.084	U	0.059	P	0.030		
				W	0.009		

a) Above is the Huffman tree for the Hawaiian alphabet. Using the tree and the labeled lines, write the bit code for each letter in the table below.

Letter	Code	Letter	Code
ʻ	0111	l	0000
a	10	m	11000
e	1101	n	1110
h	0001	o	010
i	1111	p	110011
k	001	w	110010
u	0110		

b) Now encode the Hawaiian words below

ʻAʻolepilikia ____

Hoaloha ____

Moʻopuna ____

9. (Classes and OOP)
   Create a class called `Worker`. *Worker* holds information on a factory worker in a company. The information includes the worker's full name, hourly rate, hours in a standard week and hours in an extended week.

   A worker earns their normal hourly rate for the number of hours in a standard week. If they work more hours, for the extra hours, they earn 1.5 times their hourly rate. Finally, if they work beyond the number of hours in the extended week, any hours over that number are paid at 2 times the hourly rate.

   The class must have an __init__ method to build the object given the worker's name, hourly rate, standard hours, and extended hours. You must also write a `calculatePay()` method that takes the number of hours worked that week and returns the amount of pay in US dollars.

   Example: If Joe Cool has a standard work week of 40 hours, an extended week of 50 hours, and wage of 18.50 per hour, his pay for 55 hours would be:

   40 * 18.50 + 10 * 18.50 * 1.5 + 5 * 18.50 * 2 = 1202.50

   So creating a *Worker* object to compute this would look like:
```
w = Worker("Joe Cool", 40, 50, 18.50)
print(w.name + " earned $" + str(w.calculatePay(55)) + " for 55
hours.")
```

   Write the class, creating the constructor and the calculatePay() method.

```
class Worker:
```

10. (Regular Expressions)

a) The following is a regular expression

```
r'\d\d-\d\d-\d\d \d\d:\d\d:\d\d\.\d\d'
```

What will the pattern match from the following text (<u>clearly underline</u> the exact text matched, if any, in each line):

```
18-08-26-08:00:00.01  Start of classes

18-09-23 08:00:00.00  Chuseok starts

18-09-26 23:59:59.99  Chuseok ends

18-12-12-18:30:00.99 - End of classes
```

b) What does the following code print?

```
import re
phone = '123-456-7890'
pattern = r'^\(\d{3}\)-\d{3}-\d{4}$'
if re.search(pattern, phone):
    print('The string matches the pattern.')
else:
    print('The string does not match.')
```

____

c) After the following code runs, what will be in the variable result?

```
Import re
line = 'the cat and dog'
result = re.sub(r'(.*)(dog)(.*)',
            r'\1mouse\3', line)
```

____

d) Given the following words:

1. abbc

2. bcadcbab

3. abcd

4. acdd

5. abcdd

For each of the following regular expressions, list the numbers of the words above that match that pattern. There may be multiple answers to each part; if so, you must list all of the correct answers to receive full credit.

ab*cd* ______________________________

(ab|bc)[abcd]*(ab|bc|cd) ______________________________

[ab]∗c∗a[cd]∗b∗[ab]∗ ______________________________

Q11 (Expressions with mod)

What does Python print as the value of the following expressions?

19 % 5	________________________
21 % 7	________________________
((21 * 7) + 16) % 31	________________________
((20 * 80) + 337) % 1000	________________________
((100 % 19) + 20) % 7	________________________
((10 * 20) % 5 + 30) % 4	________________________
17 % 2 + 31 % 2	________________________
(700 + 3) % 70	________________________

Q12 (Code analysis)

a) What does the following code print:

```
contractions = {"I'm": "I am", "You're": "You are", "He's": "He is",
"She's": "She is"}
sentences = ["I'm finished.", "You're good.", "He's there.", "She's
awesome."]
for sentence in sentences:
    words = sentence.split()
    if words[0] in contractions.keys():
        newsentence = contractions[words[0]]
        for word in words[1:]:
            newsentence = newsentence + " " + word
            print(newsentence)
```

b) What does the following code print:

```
contractions = {"I'm": "I am", "You're": "You are", "He's": "He is",
"She's": "She is"}
newcont = {contractions[key]:key for key in contractions.keys()}
print(str(newcont))
```

13: Computational thinking

Computers are being used in almost every fields in the day-to-day life. Specify any area of your interest (e.g. medicine, engineering, meteorology, banking, education, fashion, government, journalism etc.) and write a short note on how computers help solve problems in that field.

Q. 14: Computer Programming Fundamentals

A. Suppose the following strings are defined in an interactive session:

```
>>> s = "flight"
>>> t = "Delta flight en route from Detroit to Amsterdam is diverted
to Manchester."
```

What will Python print as the value of the following expressions?

```
1. len(s)            ________________________________
2. s * 3             ________________________________
3. (s + '!') * 2     ________________________________
4. s in t            ________________________________
5. t.count('t')      ________________________________
```

B. Distance between Incheon and Busan is 330 kms. Write a Python program with a function that given as input number of seconds to travel 1 km of distance, return the number of hours required to travel between Seoul and Busan (with a precision of two decimal points).

```
 >>>Enter number of seconds to travel 1 km: 70
Time required to travel from Incheon to Busan: 6.42 hours
```

Q. 15: Iteration, lists and algorithm design

A. Suppose a list is defined with this assignment statement:

```
>>> names = ['Korea', 'Japan', 'Malaysia', 'Singapore', 'Philippines']
```

What Python will print if we evaluate each of the following expressions:

```
1. len(names)                 ________________________________
2. len(names[0] + names[1])   ________________________________
3. names[2] == 'Taiwan'       ________________________________
```

Suppose that the following expression is typed in Python shell. (assuming names is defined as above).

```
>>> names.insert(1, 'Indonesia')
```

What will be the output of following:

```
4. 'Indonesia' in names ________________________________
5. names.index('Japan') ________________________________
```

B. Suppose a list is defined with this assignment statement:

```
>>> notes = ['Breve', 'Semibreve', 'Minim', 'Crotchet', 'Quaver']
```

What Python will print after executing following statements?

```
for i in range(0,len(notes)):
    print(notes[i], 'has', len(notes[i]), 'letters')
```

____

____

____

____

____

Q. 16: Iteration, lists and algorithm design

A. Write a Python program that uses function `def printAsteriskPattern(num_rows)` to construct the following pattern, using a nested for loop. The function receives as input the number of rows in a pattern.

```
>>>printAsteriskPattern(5)

*
* *
* * *
* * * *
* * * * *
```

B. Suppose two lists are defined with this assignment statements:

```
>>> list1 = ['physics', 'chemistry', 1997, 2000]
>>> list2 = [1, 2, 3, 4, 5, 6, 7]
```

What Python will print if we evaluate each of the following expressions:

1. `print ("list1[0]: ", list1[0])`

   ____

2. `print ("list1[-2]: ", list1[-2])`

   ____

3. `print ("list1[1:]: ", list1[1:])`

   ____

4. `print ("list2[1:5]: ", list2[1:5])`

   ____

Q. 17: Searching, sorting and scalability

A. Suppose a list is defined with the following assignment statement:

```
names = ['Mendeleev', 'Pascal', 'Darwin', 'Neuman', 'Galileo', 'Turing']
```

What would be the output of executing following expressions?

```
>>> from PythonLabs.IterationLab import *

1. isearch(names, 'Neuman') _____________________________

2. isearch(names, 'Turing') _____________________________

3. isearch(names, 'Newton') _____________________________
```

B. What is the time complexity of the following codes?

```
a = 0
i = N
while (i > 0):
    a += i
    i /= 2
```

1. O(N)
2. O(Sqrt(N))
3. O(N / 2)
4. O(log N)

```
a = 0
for i in range (0, N):
    for j in range (N, i, -1):
        a += 1
```

1. O(N)
2. O(N*log(N))
3. O(N * Sqrt(N))
4. O(N*N)

Q. 18: Divide and conquer, recursion

A. Estimate the number of comparisons required to search a list with n items using binary search technique.

Tip: Calculate $\log_2 n$ and round up to the nearest integer (ceiling).

1000 items	10 searches
10000 items	_______ searches
100000 items	_______ searches
1000000 items	_______ searches

B. Suppose a list is defined with the following assignment statement:

```
elems = ['Rf', 'Sn', 'Au', 'Ge', 'Bh', 'Sr', 'Cn', 'Y']
```

Sort the list using mergesort (msort) technique.

>>>msort (elems)

```
[Rf] [Sn] [Au] [Ge] [Bh] [Sr] [Cn] [Y]
```

____

____

____

C. Write a recursive function to find whether a given string is palindrome. A palindrome is a word, phrase, or sequence that reads the same backward as forward, e.g., `madam` or `dad`.

```
# Return True if the argument is a palindrome, and False if not
def is_palindrome(s):
```

Q. 19: Machine learning and string manipulation

Write a function, *acronym*, that creates an acronym from the first letter of each long word in a list, where a long word is any word with more than three letters. The acronym function takes an optional argument that tells it to include the first letter of all words, but short words (three or less letters) should not be capitalized.

```
>>> acronym('department of motor vehicles')

'DMV'

>>> acronym('department of justice', True)

'DoJ'
```

Q. 20: Object oriented programming

The class `Clock` simulates the tick-tack of a clock as it represents a second passing.. An instance of this class contains the time, which is stored in the attributes `self.hours`, `self.minutes` and `self.seconds`. Complete the `tick` method, which adds another second to the time, and `__str__` method of the Clock class.

```
    Examples:

    >>> x = Clock(12,59,59)

    >>> print(x)

    12:59:59

    >>> x.tick()

    >>> print(x)

    13:00:00

    >>> x.tick()

    >>> print(x)

    13:00:01

    """


def __init__(self, hours, minutes, seconds):
     """

     The parameters hours, minutes and seconds have to be integers and must
     satisfy the following equations: 0 <= h < 24, 0 <= m < 60, 0 <= s <
     60. An exception is thrown if the values are outside range.

      """

        if type(hours) == int and 0 <= hours and hours < 24:

            self._hours = hours

        else:

            raise TypeError("Hours have to be integers between 0 and 23!")

        if type(minutes) == int and 0 <= minutes and minutes < 60:

            self.__minutes = minutes

        else:

            raise TypeError("Minutes have to be integers between 0 and 59!")

        if type(seconds) == int and 0 <= seconds and seconds < 60:

            self.__seconds = seconds

        else:

            raise TypeError("Seconds have to be integers between 0 and 59!")
```

```
def tick(self):
      """
      This method lets the clock "tick", this means that the internal time
      will be advanced by one second.
      """
```

```
def __str__(self):
        """
       Prints the time in the format HH:MM:SS.
        """
```

Q. 21: Cryptography

A. Use Caeser Cipher technique with right shift of 3 letters (e.g. A is shifted to D) for encrypting the following message:

Plain text: B A N G A P S U M N I D A

Encrypted message: ____

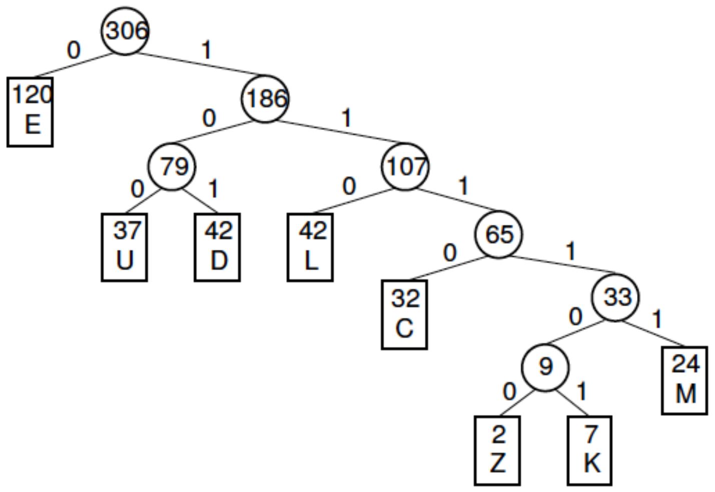
B. The message below was encoded with a Caeser Cipher technique with right shift of 3 letters (e.g. A is shifted to D). Decrypt the following message:

Encrypted message: Q L F H W R P H H W B R X

Decrypted message: ____

Q. 22: Data representation and compression

Huffman coding problem



c) Above is the Huffman tree for a certain language. Using the tree and the labeled arcs, write the bit code for each letter in the table below.

Letter	Code	Letter	Code
E		C	
U		Z	
D		K	
L		M	

d) Now encode the words below and write the number of bits required to represent the word.

DEED ____

MUCK ____

e) Now decode the strings below and identify the word.

1101001110111101 ____

11101001011011100 ____

Q. 23: Natural language processing

A. What does the following code print:

```
from PythonLabs.ElizaLab import Pattern
p = Pattern('cow|pig|horse')
p.add_response('How many $1s were on the farm?')
result = p.apply('The horse jumped the fence.')
print(str(result))
```

____

B. What does the following code print:

```
from PythonLabs.ElizaLab import Pattern
p = Pattern('I (like|love|adore) my (cat|dog|ducks)')
p.add_response('Why do you $1 your $2?')
result = p.apply('I adore my cat.')
print(str(result))
```

____

Q. 24: Regular expressions and number conversions

A. What does the following code print:

```
import re
phone = '123-456-7890'
creditcard = '4865 3456 7888 1234'
pattern = r'^\d{3}-\d{3}-\d{4}$'
if re.search(pattern, phone):
    print("There is a match.")
else:
    print("There is no match.")
if re.search(pattern, creditcard):
    print("There is a match.")
else:
    print("There is no match.")
```

____

____

B. Every Acme Software, Inc., product license number has the same basic format: 3–6 character groups, separated by single dashes. Each character group contains either 5 or 6 characters: two (upper or lowercase) letters, followed by a single digit, followed by two more (upper or lowercase) letters, followed by an optional '@' symbol.

For example, these are both valid product licenses:

```
GD5IB@-Jz5pA-MO4mT-RF4hA
sN0QO-va0Qs-Rv6uS@-bm4hx-Ka5oI@-yQ5no@
```

Define a regular expression that will match **EXACTLY** this pattern. Be careful!

Q. 25: Write a function `def dec2bin` that takes as input a decimal number and returns its equivalent in a binary number format.

```
>>>print(dec2bin(100))
1100100
>>>print(dec2bin(123))
1111011
```

REFERENCE:

The final exam will include this table for reference on regular expressions:

Operator	Means	Character Class	Matches
$*$	0 or more occurrences	[ ]	Indicates a generic character class
$+$	1 or more occurrences	\d	Any single digit (0–9)
?	0 or 1 occurrence	\D	Any single non-digit character
$\{n\}$	Exactly n occurrences	\w	A single alphanumeric character (a–z, A–Z, 0–9, or an underscore)
$\{m, n\}$	At least m and no more than n occurrences	\W	A single non-alphanumeric character
$\{m, \}$	At least m occurrences	\s	Any single whitespace character (space, tab, or newline)
( )	Marks a group or sub-pattern	\S	A single non-whitespace character
|	Indicates alternation (OR)		