CSE101 – Fall 2019
Final Exam: December 12, 2019 (100 points)

Name: _____   ID No: _____

Instructions: Read the questions carefully before attempting to write the answer. Please write your answers neatly in the space provided below each question. We encourage you to use a pencil, so that you can erase if needed.

Please **choose THREE of the following four problems: #10, #11, #12, or #13** to complete. You may skip ONE of those problems. If you have time and wish to complete all problems, I will grade all problems and give you the highest three scores.

Please **choose TWO of the following three problems: #14, #15, or #16** to complete. You may skip ONE of those problems. If you have time and wish to complete all problems, I will grade all problems and give you the highest two scores.

The last page of the exam contains some helpful reference information on regular expressions and the English alphabet, as well as scratch paper. Feel free to detach that page for easy reference while taking the exam. Only use the scratch paper for working through problems, not writing answers.

Question #	Points Available	Points Earned	
1	8		
2	10		
3	7		
4	4		
5	4		
6	11		
7	5		
8	8		
9	8		
10	5		
11	5		
12	5		
13	5		
14	10		
15	10		
16	10		
Total:	100		

1. a. (1 point) Convert `2B` from base 16 to base 10.

   b. (1 point) Internet speed is sometimes labeled in "megabits per second". 12 megabits per second is equal to how many megabytes per second?

   c. (2 points) What will be displayed by the following Python code?
   ```
   x = 201
   if x > 200:
       print("x is greater than 200")
   if x > 100:
       print("x is greater than 100")
   ```
   ```
   a.  x is greater than 100
   b.  x is greater than 100   followed by x is greater than 200
   c.  x is greater than 200
   d.  x is greater than 200   followed by x is greater than 100
   e.  Nothing will be displayed
   ```

   d. (2 points) Given the following strings, what Python code would create the string "Hello there Alice"?
   ```
   string = "Hello there Bob"
   other = "My name is Alice"
   ```
   ```
   a. string[11:] + other[:11]
   b. string[:11] + other[12:]
   c. string[:12] + other[11:]
   d. string[:12] + other[:11]
   e. None of the above
   ```

   e. (2 points) What is the result of encoding the word "home" using a Caesar Cipher with a right shift of 6 (e.g. A -> G)?
   ```
   a. mrtj
   b. nusk
   c. xecu
   d. bigy
   e. None of the above
   ```

2. (10 points) What will be the output for the following Python statements?

```python
elems = ['Yb', 'Ge', 'B', 'As', 'Ac', 'Pt', 'Ir', 'Ra']


print(elems[2] * 3)              _____


print(elems[-1] + elems[1])      _____


print(elems[2:7:2])              _____


elems.sort()
# The following operations are performed with the sorted elems list

elems.insert(3, 'C')
elems.remove('Ge')
print(elems)                     _____


elems.extend(['Li', 'Be', 'B', 'Si'])


print(len(elems))                _____
```

3. a. (2 points) In 1-2 sentences, briefly describe how symmetric key encryption works.

    b. (2 points) What is a major problem with symmetric key encryption and how is that problem solved with modern cryptography techniques?

    c. (1 point) What is one task that a computer solves really well?

d. (2 point) What are 2 tasks that a computer is unable to do or solve currently?

4. Consider the following Python program:

a. (2 points) What is the output of the program?

```
def mystery(n):
    total = 0
    for i in range(n):
        a = n
        while a > 0:
            total += a
            a //= 2
    return total

print(mystery(4))    _____
```

b. (2 points) What is the worst-case time complexity of the `mystery` function?

   a. O(logn)
   b. O(n)
   c. O(nlogn)
   d. O(n^2)

5. Consider the following Python program:

a. (2 points) What is the output of the program?

```
def other_mystery(n):
    list = []
    for x in n:
        if x % 2 == 0:
            list.append(x)
        else:
            list.append(x*2)
    return list

print(other_mystery([1,2,3,4,5,6]))  _____
```

b. (2 points) What is the worst-case time complexity of the `other_mystery` function?

   a. O(logn)
   b. O(n)
   c. O(nlogn)
   d. O(n^2)

6. a. (4 points) Suppose several lists are defined with the following assignment statements:

```
names1 = ['Lovelace', 'Neumann', 'Knuth', 'Hopper', 'Liskov']
names2 = ['Dijkstra', 'Ritchie', 'Shannon', 'Brooks', 'Allen']
names3 = names1 + names2
```

How many comparisons will be made by the following searches using linear search?

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

1. isearch(names1, 'Knuth') _____

2. isearch(names2, 'Brooks') _____

3. isearch(names3, 'Allen') _____

4. isearch(names3, 'Gates') _____

b. (3 points) Consider the following list of numbers:

```
[4, 15, 21, 26, 32, 45, 61]
```

List ALL of the values that must be examined, in the order you examine them, if you use binary search to find

the value 32 in the list above? _____

c. (2 points) Which of the following strings correctly matches the following regular expression?
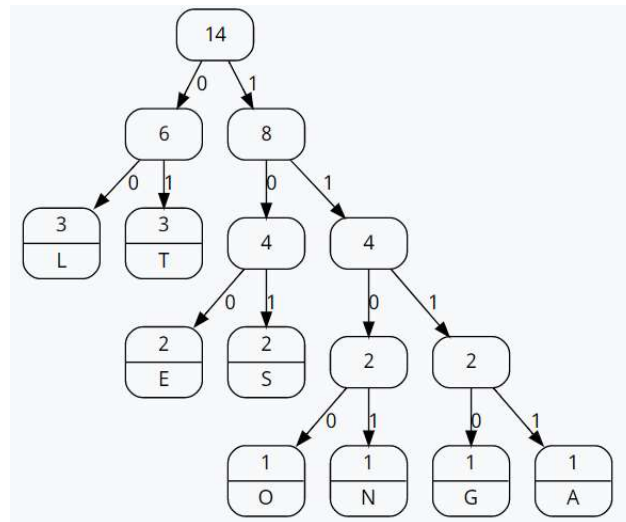
```
\d+/\d+/\d{4}
```

a. 1/4/17
b. 12/09/2018
c. 11\15\2019
d. Sept/23/2016
e. None of the above

d. (2 points) Which regular expression below will correctly match the following string?

```
-9.37E21
```

a. [\d]+\.\dE[\d]+
b. [\d]+\.[\d]+E\d
c. [+-][\d]+\.[\d]+E[\d]+
d. [+-]\d\.\dE\d
e. None of the above

(5 points) Answer the following problems using the below Huffman tree for a language.

a) (1 point) Using the above Huffman tree, write the bit code for each letter in the table below.

Letter	Code	Letter	Code
L	00	O	1100
T	01	N	1101
E	100	G	1110
S	101	A	1111

b) (2 points) Now encode the word below **AND** write the number of bits required to represent the encoded word.

LONGEST   ___00 1100 1101 1110 100 101 01 = 22 bits___

c) (2 points) Now decode the string below and identify the word.

00100111110101   ___LEAST___

8. a. (4 points) What does the following code print?

```python
contractions = {"can't": 'can not', "don't": 'do not', "won't": 'will not',
"you're": 'you are'}
sentences = ["It can't happen!", "Please, don't litter!", "won't you help
me?", "You're great!"]


for sentence in sentences:
    words = sentence.lower().split()
    if words[0] in contractions.keys():
        new_sentence = contractions[words[0]]
        for word in words[1:]:
            new_sentence = new_sentence + " " + word
        print(new_sentence)
```

_____

_____

_____

_____

b. (2 points) What does the following code print:

```python
from PythonLabs.ElizaLab import Pattern
p = Pattern('tiger|wolf|elephant')
p.add_response('How many $1s were at the zoo?')
result = p.apply('I had an elephant ride today.')
print(str(result))
```

_____

c. (2 points) What does the following code print:

```python
from PythonLabs.ElizaLab import Pattern
p = Pattern('I (walk|bike|skate) to the (school|gym|home)')
p.add_response('Why do you $1 to the $2?')
result = p.apply('I bike to the gym.')
print(str(result))
```

_____

9. Each of the following Python code examples contains a bug. Write the line number containing the bug and then show how to fix it. You may only change **one** line to make the fix.

A. (4 points) The `date_decoder()` function converts a string containing a date with the format Day-Month-Year to a tuple containing year, month, and day as separate integers. The month is input as a 3-character uppercase string (e.g. JAN for January) and the year is input as 2-digits. If the year is less than 50 it assumes the year is in the 2000s, otherwise if the year is >= 50 it assumes the year is in the 1900s.
For example:
`date_decoder('12-MAR-85')` should return (1985,3,12) and
`date_decoder('2-JUN-49')` should return (2049, 6, 2)

```
1. def date_decoder(date):
2.    months = {'jan': 1, 'feb': 2, 'mar': 3, 'apr': 4, 'may': 5, 'jun': 6,
            'jul': 7, 'aug': 8, 'sep': 9, 'oct': 10, 'nov': 11, 'dec': 12}
3.    parts = date.split('-')
4.    a = int(parts[0])
5.    b = months[parts[1]]
6.    c = 1900 + int(parts[2])
7.    if int(parts[2]) < 50:
8.        c += 100
9.    return c, b, a
```

Line _____ contains a bug.

It should be _____.

b. (4 points) The destination function takes a max distance number as an integer and returns the city that is the farthest distance away, but not farther than the max distance. The places dictionary provides a list of cities and their distance away in kilometers. If there is no city within the max distance, the function returns "nowhere".

For example: `destination(4000)` should return 'Nay Pyi Taw'
        `destination(4500)` should return 'Bishkek'

```
1.  def destination(max_distance):
2.     places = {'Busan': 330, 'Incheon': 27, 'Addis Ababa': 9230,
                'Bishkek': 4409, 'Nay Pyi Taw': 3572, 'Taipei': 1482}
3.     destination = "nowhere"
4.     m = 0
5.     for c in places:
6.         d = places[c]
7.         if d <= max_distance:
8.             destination = c
9.             m = d
10.    return destination
```

Line _____ contains a bug.

It should be _____.

10. (5 points) Write a function that determines the "numerical value" of a name provided as an input string. The value of a name is determined by summing up the values of the letters of the name where "a" is 1 "b" is 2 "c" is 3 up to "z" being 26. For example, the name "Zelle" would have the value 26 + 5 + 12 + 12 + 5 = 60. The input string is case insensitive. For example, "Zelle" and "zELLE" will provide the same output.
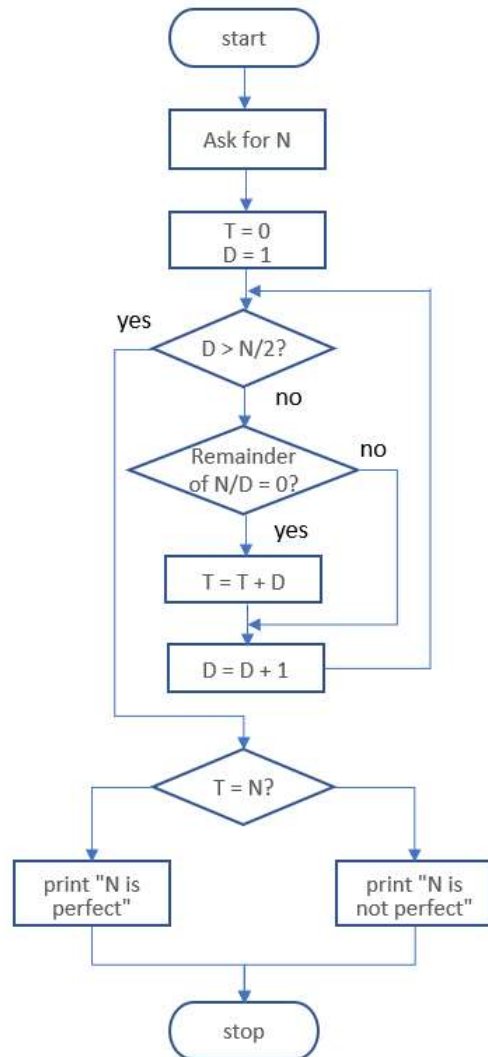
```
def numero(input):
```

11. (5 points) Write a function that takes a number of lines as input and prints a triangular pattern of ASCII characters, starting with "A" on the first line, and continuing to print the next ASCII characters on each following line. For example, given input 5, the function prints the following pattern:

```
A
B C
D E F
G H I J
K L M N O
```

```
def asciipyramid(num):
```

12. (5 points) Given the following flowchart to check whether a number is a perfect number, write the equivalent Python code. A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. E.g. 6 has divisors 1, 2 and 3 (excluding itself), and 1 + 2 + 3 = 6, so 6 is a perfect number.

13. (5 points) Create a regular expression that will match any undergraduate course code at a university. All courses are in the form of any three capitalized letters, followed by three digits. The first digit always starts with the number 1,2,3, or 4 for an undergraduate course.

For example, these are all valid matches:

```
CSE101
CSE336
AMS480
BME296
```

But it should not match:
```
CSE614
cse101
CSE40
CS101
```

_____

14. (10 points) Complete the `retrieve()` function below, which uses recursion to retrieve and return the value at index `n` from a Python list. You may assume that the list always contains at least `n+1` elements. Your implementation **MUST** solve the problem using recursion in order to receive **ANY** credit for this problem!

HINT: If `n` is `0`, return the first element of the list (index 0); otherwise, return the value at index `n-1` of the rest of the list (everything after the first element).

```
def retrieve(n, list):
```

15. (10 points) Complete the `weight_of_molecule()` function below which computes the molecular weight of a molecule that can contain only hydrogen (H), carbon (C), or oxygen (O) atoms. The function takes a chemical formula such as H2O or HCO3 as input and returns the total molecular weight of all the atoms based on these individual atom weights:

Atom	H	C	O
Weight (grams/mole)	1.00794	12.0107	15.9994

For example, the molecular weight of water (H2O) is: 2*(1.00794) + 15.9994 = 18.01528.
The molecular weight of bicarbonate (HCO3) is: 1.00794 + 12.0107 + 3*(15.9994) = 61.0168
So `weight_of_molecule('HCO3')` returns `61.0168`

Every atom is a single character (e.g. H) and may be followed by 1 digit to represent the quantity of that atom (e.g. H3), otherwise the quantity is 1. Hint: It may be helpful to use the `isdigit` method on a string to check whether a string consists of only digits – for example `"2".isdigit()` returns True.

```
def weight_of_molecule(input):
```

16. (10 points) Fill out the `Profile` class below, following the instructions given to get the expected output. The `Profile` class is intended to store a person's online Profile and is initialized with that person's first name, last name, and the year they were born, and contains some additional methods to print the person's name and initials, as well as determine their zodiac animal. For this problem you can assume that the `Profile` is always initialized with valid information and that the Zodiac animals match up perfectly with the year the user was born.

```python
class Profile:

    zodiac_dict = { 0: "Monkey", 1: "Rooster", 2: "Dog", 3: "Pig", 4: "Rat", 5: "Ox",
        6: "Tiger", 7: "Rabbit", 8: "Dragon", 9: "Snake", 10: "Horse", 11: "Goat" }

    # Create three instance variables to store the input parameters
    def __init__(self, first_name, last_name, year_born):




    # When a Profile is printed, it should display the person's first and last name,
    # separated by a space
    def __repr__(self):




    # Returns the person's capitalized initials (first letter of their first and last name)
    def get_initials(self):




    # This returns the zodiac animal for the year they were born. Use the zodiac_dict above,
    # which tells the animal for Year 0-11. The animals repeat every 12 years, so people born
    # in year 0,12,24…1980,1992, and 2004 would all be "Monkey"
    def get_zodiac_animal(self):




# Below is the code using the complete Profile class and the expected output
p = Profile("Alice", "Kim", 1996)
print(p)                      # should print "Alice Kim"
print(p.get_initials())       # should print "AK"
print(p.get_zodiac_animal())  # should print "Rat"
```

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)		

English Alphabet (for the Caesar cipher):

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Scratch Paper