CS150 Fall 2023 – Midterm 2 Solution

Name: _____ **Section: A**

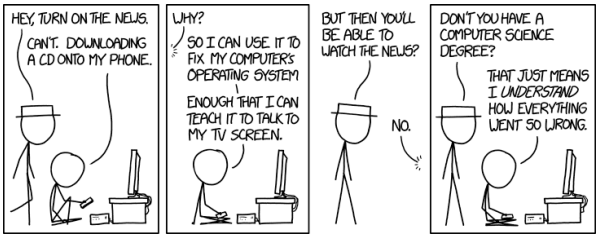
Date: _____ **Start time:** _____ **End time:** _____

Honor Code:

Signature: _____

This exam is closed book, closed notes, closed computer, closed calculator, etc. You may only use (1) the midterm 1 "cheat sheet" from the course page, and (2) a single double-sided letter sheet of notes of your own creation. **You have 2.5 hours.** Read the problem descriptions carefully and write your answers clearly and *legibly* in the space provided. Circle or otherwise indicate your answer if it might not be easily identified. You may use extra sheets of paper, stapled to your exam, if you need more room, as long as the problem number is clearly labeled and your name is on the paper. If you attached extra sheets indicate on your main exam paper to look for the extra sheets for that problem.

You <u>do</u> need to include module imports (if relevant for your code), but <u>do not</u> need to include comments, docstrings or constants in your code.



Question 9: Implications of the Python memory model

(a) Assume that `x` is a non-empty list that could contain integers and lists of integers. Which of the following code snippets would *change* the value of `x`? *Select all that apply.*

√
```
y = x
y.append(2)
```

◯
```
y = x[:]
y.append(2)
```

◯
```
y = x + [2]
```

(b) "Aliasing" occurs when there is more than one variable name referencing the same object in memory. None of the answers above are sufficient to prevent all forms of aliasing between `x` and `y`. Complete the program below (writing relevant code on the lines) by providing an initial value for `x`, and a set of operations performed exclusively on `y` that would *change* the value pointed to by `x`, i.e., after the code executes `x` is not equal to its initial value. Briefly explain your answer. As above, `x` should be an non-empty list that can contain integers and lists of integers. You may not need all the lines.

```
# Define a value for x

x = ____[1, [2, 3], 4]____
y = x[:]
# Operations only on y that would change the value of x

____y[1].append(5)____

_____________________________________
```

Solution: Slicing only makes a copy "one level" deep, that is the nested list is still aliased. Thus modifying the nested list through `y` will modify the value pointed to by `y`.

Question 10: Applications of data structures

(a) For each of the following applications, indicate whether a list, set, dictionary, or tuple would be the *most* appropriate data structure to use.

i. Storing the unique collection of letters guessed so far in a word guessing game.
◯ List √ **Set** ◯ Dictionary ◯ Tuple

ii. Storing all the official marathon times under 2 hours and 10 minutes in ascending order of time.
√ **List** ◯ Set ◯ Dictionary ◯ Tuple

iii. Storing the collection of road names in Addison county and their corresponding length in miles.
◯ List ◯ Set √ **Dictionary** ◯ Tuple

iv. Storing the latitude and longitude of a location.
◯ List ◯ Set ◯ Dictionary √ **Tuple**

(b) Python supports multiple character encodings, that is mappings between characters and integers. Imagine a encoding for letters where the characters "a-zA-Z" map to the integers 0-51, inclusive (i.e., "a" is encoded as 0, "A" as 26, ...).

i. `D` is a Python object such that, if `i` is an integer between 0 and 51, then `D[i]` is the letter for that encoding value. For example, `D[1]` evaluates to `"b"` and `D[26]` to `"A"`. Which of the following could be the type of `D`? *Select all that apply.*

√ **A string**
√ **A list**
◯ A set
√ **A dictionary**
√ **A tuple**

ii. `E` is a Python object such that, if `c` is a single character then `E[c]` is the integer encoding for `c`. For example, `E["d"]` evaluates to `3`. Which of the following could be the type of `E`? *Select all that apply.*

◯ A string
◯ A list
◯ A set
√ **A dictionary**
◯ A tuple

Question 11: Writing functions with sets

I typically teach two classes per semester. I want to know how many students are in only one of my classes, and how many are in both. Write a function named `enroll` that takes two lists of names and returns a tuple with number of students in only one class, and the number in both. Your solution will be evaluated based on the correctness, efficiency and conciseness of your approach. For example:

```
>>> class1 = ["Hopper", "Church", "Babbage", "Turing"]
>>> class2 = ["Wolfram", "Simon", "Hopper", "Babbage", "Neumann"]
>>> enroll(class1, class2)
(5, 2)
```

Solution:

```
def enroll(class1, class2):
    set1 = set(class1)
    set2 = set(class2)
    return (len(set1 ^ set2), len(set1 & set2))
```

Question 12: Writing functions with dictionaries

One of the weaknesses of our substitution encryption function is that it does not obscure the frequency of letters in the original message. English has a very characteristic letter frequency that can be used to "break" the encryption. For example, the most common letter in the encrypted message is likely an encrypted "e" (the most common letter in English text). To assist in breaking a substitution encryption, write a function named `letter_freq` that takes a string as its argument and returns a dictionary with the frequency (as a fraction) of each letter in the string. Your solution will be exclusively evaluated on correctness. For example:

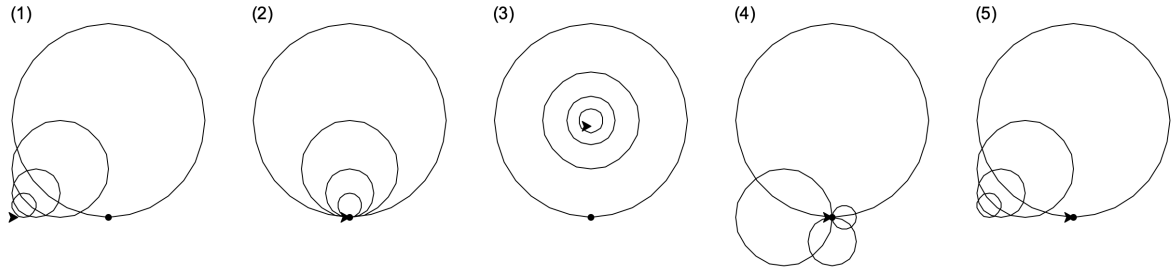
```
 >>> letter_freq("abc")
{'a': 0.33, 'b': 0.33, 'c': 0.33}
>>> letter_freq("aabc")
{'a': 0.5, 'b': 0.25, 'c': 0.25}
```

Solution:

```
def letter_freq(string):
    freq = {}
    for char in string:
        if char in freq:
            freq[char] += 1
        else:
            freq[char] = 1
    for char in freq:
        freq[char] /= len(string)
    return freq
```

Question 13: Understanding and using recursive functions

(a) For each of the 4 functions `funA - funD` defined below, identify the resulting picture by number among the 5 options below. Assume in each case the function was called like "`funX(100,4)`", e.g., `funA(100,4)`. The turtle starts at the origin (marked with a dot) facing to the right and ends at the position and heading shown by the arrow.



2

```
def funA(s, n):
    if n > 0:
        t.circle(s)
        funA(s/2, n-1)
```

4

```
def funB(s, n):
if n > 0:
    t.circle(s)
    t.left(90)
    funB(s/2, n-1)
```

3

```
def funC(s, n):
    if n > 0:
        t.circle(s)
        t.penup()
        t.left(90)
        t.forward(s/2)
        t.right(90)
        t.pendown()
        funC(s/2, n-1)
```

5

```
def funD(s, n):
    if n > 0:
        t.circle(s)
        t.penup()
        t.backward(s/2)
        t.pendown()
        funD(s/2, n-1)
        t.penup()
        t.forward(s/2)
        t.pendown()
```

(b) For the remaining 5th picture, provide the missing function definition below. Your function should be named `funE`. Write the corresponding picture number on the line below.

1

Solution:

```
def funE(s, n):
    if n > 0:
        t.circle(s)
        t.penup()
        t.backward(s/2)
        t.pendown()
        funE(s/2, n-1)
```

Question 14: Finding errors (in recursive functions)

The following recursive function is supposed to return a copy of a list with any items within the range $[begin, end]$, inclusive, removed. Unfortunately, there are several different problems with this function. Identify (by line number), explain, and fix all of the problems, and rewrite the corrected function, still using a recursive strategy. The errors should not be variations of the same issue and should impact correctness, not just style. Examples of the intended behavior are shown below.

```
>>> drop_range([-1, 1, 2, 1], 0, 1)
[-1, 2]
>>> drop_range([1, 2, 1], 0, 4)
[]
>>> drop_range([], 0, 1)
[]
```

```
1 def drop_range(a_list, begin, end):
2     if a_list == []:
3         return []
4     if begin <= a_list[0] <= end:
5         return [a_list[0]] + drop_range(a_list, begin, end)
6     else:
7         return drop_range(a_list, begin, end)
```

Solution:

1. Recursive cases (lines 5 and 7) are reversed.
2. The recursive calls on lines 5 and 7 should be on `a_list[1:]`, not `a_list`. THe input is not getting smaller and so the recursion will be infinite.

```
def drop_range(a_list, begin, end):
    if a_list == []:
        return []
    if begin <= a_list[0] <= end:
        return drop_range(a_list[1:], begin, end)
    else:
        return [a_list[0]] + drop_range(a_list[1:], begin, end)
```

The use of a second `if` on line 4 is stylistic choice, but not an error in this context as the `return` statement on line 3 will terminate the function if the `if` statement on line 2 is true.

Question 15: Using Object-Oriented Programming

Consider the following classes for describing convex 2-D shapes.

```
class Polygon:
    def __init__(self, sides):
        """Create a convex Polygon

        Args:
        sides: List of side lengths in
          clockwise order, e.g. [1, 1, 1]
          for an equilateral triangle
        """
        self.sides = sides

    def num_sides(self):
        """Return number of sides"""
        return len(self.sides)
```

```
class EquilateralTriangle(Polygon):
    def __init__(self, side):
        super().__init__([side] * 3)

class Square(Polygon):
    def __init__(self, side):
        super().__init__([side] * 4)

class Rectangle(Polygon):
    def __init__(self, side1, side2):
        super().__init__([side1, side2]*2)
```

(a) Write a `perimeter` method that returns the perimeter of a shape, e.g., `Square(4).perimeter()` should evaluate to 16. As a hint, you should be able to write one method that works for `EquilateralTriangle`, `Square` and `Rectangle`. You only need to provide the method itself, not the surrounding class(es). Your solution will be evaluated based on the correctness, efficiency and conciseness of your approach.

Solution:

```
def perimeter(self):
    return sum(self.sides)
```

(b) In which classes will you need to implement the `perimeter` method to be able to compute the perimeter for all of the shapes? *Select all that apply.*

- √ `Polygon`
- ◯ `EquilateralTriangle`
- ◯ `Square`
- ◯ `Rectangle`

(c) One of the proposed features for your program is "Draw the shape using Turtle". For the feature, "draw" would best map to a:

- ◯ Class
- √ **Method**
- ◯ Instance variable
- ◯ Initializer
- ◯ Parameter

Question 16: Vectorized execution

Translate each of the code blocks below to just use Python built-in functions and operators, assuming that `x` and `y` are non-empty lists of floats of the same length (instead of a NumPy vector). If the function returns a vector, your function should return a list. The functions `min`, `max`, `sum` and the `math` module are considered built-in Python. Your solution will be evaluated based on the correctness, efficiency and conciseness of your approach.

(a)
```
def mystery(x, y):
    return np.max(x) > 0 and np.max(y) > 0
```

Solution:
```
def mystery(x, y):
    return max(x) > 0 and max(y) > 0
```

(b)
```
def mystery(x):
    return np.sqrt(np.sum(x * x))
```

Solution:
```
import math
def mystery(x):
    result = 0
    for val in x:
        result += val * val
    return math.sqrt(result)
```

Page intentionally left blank.