Q1 Welcome!

0 Points

Welcome to Part 2 of the Final Exam for CSE 160 20au! Have fun!
Don't forget to also submit Part 1 (function writing)!

Point values on Part 1 and Part 2 are not final and may be adjusted slightly.

(Optional - Just for fun!) If you were designing your own programming language, what animal would YOU name it after and why?

Q2 Mystery

4 Points

Given the following function, what **input** would *return*:
`[0, 2, 8, 5, 5, 5, 10, 20, 3]` ?

```
def mystery(x):
    result = []
    for i in x:
        if i % 2 == 0:
            result.append(i)
        else:
            for j in range(i):
                result.append(i + 2)
    return result
```

That is, give a possible value for `var` if the call:

```
mystery(var)
```

returns:

```
[0, 2, 8, 5, 5, 5, 10, 20, 3]
```

Give a possible value for `var` that would lead to the value above being returned:

[0, 2, 8, 3, 10, 20, 1]

Q3 A bug and a fix

4 Points

Consider the following function:

```
def get_index(lst, x):
    '''
    Given a list called lst, returns the index of the first occurrence
    of x in lst. Returns -1 if x is not in lst.
    '''
    for i in range(len(lst)):
        if lst[i] == x:
            return x
```

Q3.1 A bug!

2 Points

The function above contains a bug. Write a test case that will demonstrate the bug. Write your test case in the form of an assert statement.

There are actually two bugs; (you only needed to mention one of t
hese)
1) If x is not in lst, then None will be returned instead of -1:
assert get_index([1, 2, 3], 5) == -1
2) The function does not return the INDEX of the first occurrence o
f x (instead it returns x):
assert get_index([4, 5, 6], 5) == 1

Q3.2 Fix it!

2 Points

Re-write the function `get_index` to fix this bug. Feel free to cut and paste the code from above and edit it.

```
def get_index(lst, x):
    '''
    Given a list called lst, returns the index of the first occurrence
    of x in lst. Returns -1 if x is not in lst.
    '''
    for i in range(len(lst)):
        if lst[i] == x:
            return i
    return -1
```

Q4 Efficiency

6 Points

The following code outputs the correct result, however, it does so very inefficiently.

```
def file_search(file_name, terms):
    '''
    Given a file name, and list of words called terms,
    Return a dictionary that maps each word in terms to
    a list of strings containing all of the lines where that
    word occurs.
    '''
    final_result = dict()
    for term in terms:
        file_to_read = open(file_name, 'r')
        for line in file_to_read:
            words_in_line = line.split()
            for word in words_in_line:
                if word in terms and term == word:
                    next_list = []
                    if word in final_result:
                        next_list = final_result[word]
                    next_list.append(line)
                    final_result[term] = next_list
    return final_result
```

List **two** significant inefficiencies in this program, explain why they are inefficient, and how you would fix these problems.

First Inefficiency, why inefficient, how you would fix:

Repeatedly reads the entire file. Reading the entire file is an expen
sive operation that we don't want to repeat. Instead of reading the
 file multiple times, we should read it in once and store it in a data
 structure so that we don't have to re-read it. Since we want to kee
p the file divided into lines, we might want to read in the file a line
 at a time and store it as a list of strings.

Second Inefficiency, why inefficient, how you would fix:

Iterates through the terms inefficiently. We iterate through each of
 the terms in our outer-most loop, but we don't need to do this to g
et correct output. If we remove this loop and the code 'and word =
= term', we would still have a correct program without the added c
omplexity.

This function also doesn't close the file after you are done reading
 it. Having a file open takes up resources that our program may nee
d. We should close the file when we are done with it.

Q5 Data Structures

6 Points

In this course, we have talked about many data structures. In particular, we have discussed lists, sets, dictionaries, tuples, and nested structures using one or more of the previous types. Listed below are a few program ideas. For each situation, give the data structure you would use in that situation and then explain WHY you would use that data structure. **No credit given without an explanation of WHY.**

Q5.1 Birthdays

2 Points

A program that keeps track of people and their birthdays

Data Structure you would use and why:

Dictionary. Since we need to associate a name with a birthday, it w
ould make sense to store a dictionary mapping names to birthdays.

Q5.2 Students
2 Points

Keeps track of the names of all of the students in this course. This program allows users to add new students and remove dropped students.

Data Structure you would use and why:

A List/Set. If we assume that student names are unique, then a set
would work. Otherwise a list would work. We have not mentioned
two values that need to be kept together, so a dictionary does not
 make sense. And a tuple is immutable so that would not allow us t
o add or remove students.

Q5.3 Unique words
2 Points

Search through a file and identify all of the unique words that show up in the file

Data Structure you would use and why:

Set. Since we want to keep track of the unique words, we want to
 use a set since it doesn't allow for duplicate items in it.

Q6 Classes
9 Points

You are given the following class:

```
class Aquarium:
    def __init__(self, size):
        """
        Creates a new, empty Aquarium of the given size (in gallons)
        """
        self.size = size
        self.fish = {}

    def add_fish(self, new_fish):
        """
        Adds a single fish to the Aquarium
        """
        self.fish[new_fish] = self.fish.get(new_fish, 0) + 1
```

Q6.1 fish per gallon

5 Points

Fill in the following method that will be added inside the `Aquarium` class.

```
def fish_per_gallon(self):
    """
    Returns the fish per gallon in this
    Aquarium: number of fish divided by number
    of gallons
    """
```

Write your code here.

```
    fish_count = sum(self.fish.values())
    return fish_count / self.size
```

Q6.2 Make an aquarium

2 Points

Write code that will create an instance of the `Aquarium` class with a size of 60 gallons and assign it to the variable `ruths_aquarium`. This code would appear outside of the class, in a client program.

```
ruths_aquarium = Aquarium(60)
```

Q6.3 Add some fish

2 Points

Write code that will add a 'goldfish' to `ruths_aquarium`. This code would appear outside of the class, in a client program, after the code you have written for Problem 6.2.

```
ruths_aquarium.add_fish('goldfish')
```

Q7 List Comprehensions

10 Points

These two questions refer to this list;

```
lst = [[1, 2, 3, 5], [2, 3, 2], [1, 1, 2, 5, 1], [], [5], [0, 34, 5]]
```

Q7.1 Just 5

5 Points

Write a list comprehension expression that would produce a list containing just the sublists in `lst` that contain the value 5. The result should be:

```
[[1, 2, 3, 5], [1, 1, 2, 5, 1], [5], [0, 34, 5]]
```

```
[li for li in lst if 5 in li]
```

Q7.2 The max

5 Points

Write a list comprehension expression that would produce a list containing just the maximum value in each sublist in `lst`. You may use the function `max` for this question. The result should be:

```
[5, 3, 5, 5, 34]
```

[max(li) for li in lst if not li == []]

Q8 You're done!

1 Point

Did you work on this Exam alone or collaborate with others? If you collaborated with others (for either part 1 or Part 2), who did you collaborate with? **Please list full names and UWNetIDs of everyone you collaborated with.**

How long did you personally spend on this exam (Part 1 + Part 2)?

CONGRATULATIONS! You have now completed CSE 160 20au! Have a wonderful Winter break! Stay Safe!

The CSE 160 Staff

Final Exam Part 2

● GRADED

46 MINUTES LATE

STUDENT

Unknown Student (removed from roster?)

TOTAL POINTS

40 / 40 pts

QUESTION 1		
Welcome!		**0** / 0 pts
QUESTION 2		
Mystery		**4** / 4 pts
QUESTION 3		
A bug and a fix		**4** / 4 pts
3.1	A bug!	**2** / 2 pts
3.2	Fix it!	**2** / 2 pts
QUESTION 4		
Efficiency		**6** / 6 pts
QUESTION 5		
Data Structures		**6** / 6 pts
5.1	Birthdays	**2** / 2 pts
5.2	Students	**2** / 2 pts
5.3	Unique words	**2** / 2 pts
QUESTION 6		
Classes		**9** / 9 pts
6.1	fish per gallon	**5** / 5 pts
6.2	Make an aquarium	**2** / 2 pts
6.3	Add some fish	**2** / 2 pts
QUESTION 7		
List Comprehensions		**10** / 10 pts
7.1	Just 5	**5** / 5 pts
7.2	The max	**5** / 5 pts
QUESTION 8		
You're done!		**1** / 1 pt