Name: _____

Email address (UW NetID): _____

# CSE 160 Winter 2016: Final Exam

(closed book, closed notes, no calculators)

**Instructions:** This exam is closed book, closed notes. You have 50 minutes to complete it. It contains 7 questions and 8 pages (including this one), totaling 70 points. Before you start, please check your copy to make sure it is complete. After the 8 pages of the exam, there is a syntax cheat sheet that you may remove. Turn in all 8 pages of the exam, together, when you are finished. When time has been called you must put down your pencil and stop writing. **Points will be deducted from your score if you are writing after time has been called.**

Total: 70 points. Time: 50 minutes.

| Problem | Max Points | Score |
|---|---|---|
| **1** | **4** | |
| **2** | **3** | |
| **3** | **4** | |
| **4** | **14** | |
| **5** | **20** | |
| **6** | **15** | |
| **7** | **10** | |
| Total | **70** | |

1) [4 pts]

a) Give two examples of a **mutable type**:

b) Give two examples of an **immutable type**:

2) [3 pts] Write code that would produce an "`IndexError`" error

3) [4 pts] Write the output of the code below in the box here:

```
sum = 0
for x in range(2, 8):
    if x % 2 == 0:
        for y in range(x):
            sum = sum + y
print 'sum:', sum
```

**MY ANSWER:**

4) [14 pts] Given a list of lists of integers, fill in the function `index_of_max_unique` below to return the index indicating which sub-list has the most unique values. For example:

**`index_of_max_unique([[1, 3, 3], [12, 4, 12, 7, 4, 4], [41, 2, 4, 7, 1, 12]])`** would return 2 since the sub-list at index 2 has the most unique values in it (6 unique values).

**`index_of_max_unique([[4, 5], [12]])`** would return 0 since the sub-list at index 0 has the most unique values in it (2 unique values).

You can assume that neither the `list_of_lists` nor any of its sub-lists will be empty. If there is a tie for the max number of unique values between two sub-lists, return the index of the first sub-list encountered (when reading left to right) that has the most unique values.

```
def index_of_max_unique(list_of_lists):
    # Your code here
```

5) [20 pts total] a) [10 pts] Write a function called `read_zoo_animals(filename)` that takes the name of a file as a parameter. You can assume that the given file contains lines of the form: `zoo_name animal` where each `zoo_name` and `animal` are strings that contain no spaces or punctuation and are separated by a single space. Here are the contents of a sample input file:

```
point_defiance monkey
national panda
woodland_park bear
national elephant
national panda
```

Your function should read in the given file and return a dictionary mapping each `zoo_name` to a list of the animals the zoo has. For a given zoo, the animals in that zoo's list should appear in the order they appear in the file. It is o.k. to have duplicate animals in the list for a given zoo. You may assume the file name provided is valid and that the file is formatted as described and includes at least one `zoo_name animal` pair. Calling `read_zoo_animals` on the file above would return this dictionary:

```
{'point_defiance': ['monkey'],
 'national': ['panda', 'elephant', 'panda'],
 'woodland_park': ['bear']}
```

```
def read_zoo_animals(filename):
    # Your code here
```

5) (cont.) b) [10 pts] Write code in the main function that will call the `read_zoo_animals` function written in part a) to open a file called "`local_zoos.txt`" and print the results in EXACTLY the following format. For the sample input shown in problem 5a) the output would be:

```
point_defiance:
 monkey
national:
 elephant
 panda
 panda
woodland_park:
 bear
```

`zoo_names` may be printed in any order, but each zoo's list of `animals` should be printed in **alphabetical order**. Don't forget the colon at the end of each `zoo_name` and a single space at the beginning of each `animal` indenting it.

```
def main():
    # Your code here
```

6) [15 pts] Note: THIS PROBLEM IS NOT RELATED TO PROBLEM 5!!

You are given the following class definition:

```
class Zoo:
    def __init__(self, zoo_name):
        '''zoo_name: a string representing name of zoo'''
        self.name = zoo_name
        self.animals = []

    def add(self, animal):
        '''animal: a string representing an animal'''
        self.animals.append(animal)
```

a) Write the code for the method below that is also a part of the class `Zoo`:

```
    def release_animal(self, free_animal):
        '''Remove all occurrences of free_animal from this zoo.
        free_animal: a string representing an animal.
        Returns the number of occurrences that were removed.
        The free_animal string must match exactly (e.g. same
        case) with the name of an animal currently in the zoo
        in order for that animal to be removed. '''
        # Your code here
```

6) (continued)

b) Write the code for the method below that is also a part of the class `Zoo`:

```
def get_num_animals(self):
    '''Returns the total number of animals in the zoo as an
    integer.'''
    # Your code here:
```

c) Write code in the `main` function to add a lion and a tiger to the `seattle_zoo`. This code is outside of the class Zoo.

```
def main():
    seattle_zoo = Zoo("woodland_park")
    # Your code here:
```

d) Describe a change to the `Zoo` class that might cause a **client** of the Zoo class to have to modify its code.

e) Describe a change to the `Zoo` class that would NOT cause a **client** of the Zoo class to have to modify its code.

7) [10 pts] a) **Draw** the entire environment, including all active environment frames and all user-defined variables, **at the moment that the MINUS OPERATION IS performed**. Feel free to draw out the entire environment, but be sure to CLEARLY indicate what will exist at the moment the **MINUS** operation is performed (e.g. cross out frames that no longer exist).

b) When finished executing, **what is printed out by this code**? **MY ANSWER:**

c) **How many different stack frames** (environment frames) are active when the call stack is DEEPEST/LARGEST? (Hint: The global frame counts as one frame.) **MY ANSWER:**

---

```
def smile(x):
    return x + 4

def funny(x):
    y = smile(smile(x))
    return y + smile(x)

def spring(y):
    x = smile(y)
    return funny(x - y)   ← ****Draw the environment at this point****

x = 7
y = 100
print spring(smile(x)) + y
```