COGS 18 Fa23 Sec B Exam 2, In-Class

Form A.

Name:
Email:
Student ID:

This test is 9.5 pts, worth 9.5% of your final grade. There are 9 questions of varying worth. No computers. No notes. Just your pencil(s) or pen(s). 75 minutes. Eyes on your own paper—of course! The problems are not in any particular order. Skip ahead if you are stuck. You may turn in your test up front and exit quietly when you are finished.

Wait to turn the page until instructed to do so.

1. (1.5 points) Recall the mean of two numbers x and y is $\frac{x+y}{2}$.

   Fill in the blanks below so that the function computes the pair-wise mean of the two input lists. That is, the function `pairwise_mean` returns a new list where each element is the mean of the corresponding two items in the input lists. For example:

```
lst1 = [0, 10, 20]
lst2 = [1, 0,  18]

assert pairwise_mean(lst1, lst2) == [0.5, 5, 19]
```

   That is, the first item in the output list is `0.5` because that is the mean of `0` and `1`, the second item is `5` because that is the mean of `10` and `0`, and so on.

   You can assume the two input lists have the same length.

```
def pairwise_mean(lst1, lst2):

    out = ______

    for i in range(0, _____________):

        # Grab the appropriate items out of each input list

        x1 = __________

        x2 = __________

        # Compute their mean

        mean = __________________

        out.append(mean)

    return out
```

2. (1.8 points) Suppose we have a list of integers called `ints`, which can be positive or negative, and we would like to return the number of prime numbers in this list. If we run into a negative number, the function should stop there and return the count of prime numbers up to that point. If we run into the number zero, the loop should continue on to the next iteration (as zero is not a prime number). Fill in the blanks below to implement this behavior.

   You may assume the existence of a function called `is_prime` that takes a number and returns `True` or `False` depending on whether the number is prime. For example, `is_prime(3)` would return `True` and `is_prime(10)` would return `False`. *Do <u>not</u> define the* `is_prime` *function. Use it as if it already exists.*

```
def count_number_of_primes(ints):

    num_primes = 0

    for n in ints:

        # If n is a negative number, exit the loop.

        if ______________________:

            ________________

        # If n is 0, go on to the next iteration of the loop

        elif ______________________:

            _____________________

        # Check to see if n is a prime number using the
        # function is_prime, increment num_primes if so

        elif ______________________________:

            _______________________________________________

    return num_primes
```

3. (1.0 points) Consider the following directory structure:

`/`

- **`work/`**
  - **`data_analysis/`**
    - **`reports/`**
      - `final_report.docx`
      - `summary.txt`
    - **`scripts/`**
      - `run_analysis.py`
  - **`blog/`**
    - **`drafts/`**
      - `post1.md`
      - `post2.md`
    - **`images/`**
      - `header.png`

1. Suppose you are currently in the directory **`work/`**, what is the *relative* path to the file `summary.txt`?

2. What is the *absolute* path to the file `post1.md`?

3. Suppose you are currently in the directory **`images/`**, what is the *relative* path to the file `post1.md`?

4. Suppose you are in the directory **`scripts/`**, what would you put in the blank to run the `run_analysis.py` script?

```
python ____________________________________
```

5. Suppose you are in the directory **`reports/`**, what could you put in the blank to run the `run_analysis.py` script? (There are at least two possible answers. Give one.)

```
python ______________________________________________
```

4. (0.5 points) Only one thing will be printed. What will be printed?

```
x = 'hi'

def output(x):
    print(x)
    return None

x = 'hola'

def echo(x):
    output(x)
    return None

x = 'privet'
x = echo('nihao')
x = 'marhaban'
```

(a) `hi`

(b) `None`

(c) `hola`

(d) `privet`

(e) `nihao`

(f) `marhaban`

5. (1.2 points) The Collatz conjecture is a famous but simple unsolved problem in math. Starting with any integer greater than 1, repeat the following:

   - If the number is even, divide it by 2.
   - If the number is odd, multiply it by 3 and add 1.

   For example, starting with the number 3:

$$\mathbf{3} \xrightarrow{*3+1} \mathbf{10} \xrightarrow{\div 2} \mathbf{5} \xrightarrow{*3+1} \mathbf{16} \xrightarrow{\div 2} \mathbf{8} \xrightarrow{\div 2} \mathbf{4} \xrightarrow{\div 2} \mathbf{2} \xrightarrow{\div 2} \mathbf{1}$$

   The conjecture is that, no matter what number you start with, eventually you reach the number 1. It seems to be true, but nobody has proven it yet.

   Fill in the blanks so that the `collatz` function below computes and returns the number of steps before the input number `n` reaches 1. (In the example above, `collatz(3) == 7` because there were 7 steps before 3 reached 1.)

   You may assume the starting input number is at least 2.

```
def collatz(n):

    steps = 0

    while ____________________:

        if n % 2 == 0: # n is even

            n = ___________________

        else: # n is odd

            n = ____________________

        steps = ______________________

    return steps
```

6. (0.6 points) Suppose you have the following incomplete code snippet:

```
dict1 = {'A': 'red',   'B': 'yellow',
         'C': 'green', 'D': 'blue'}
lst1 = []

for thing in _____________:
    lst1.append(thing)
```

Part I. If the blank was `dict1`, what will be the final value of `lst1`?

(a) `[]`

(b) `['A', 'B', 'C', 'D']`

(c) `['A', 'red', 'B', 'yellow', 'C', 'green', 'D', 'blue']`

(d) `['red', 'yellow', 'green', 'blue']`

(e) `[('A','red'), ('B','yellow'), ('C','green'), ('D','blue')]`

Part II. If the blank was `dict1.values()`, what will be the final value of `lst1`?

(a) `[]`

(b) `['A', 'B', 'C', 'D']`

(c) `['A', 'red', 'B', 'yellow', 'C', 'green', 'D', 'blue']`

(d) `['red', 'yellow', 'green', 'blue']`

(e) `[('A','red'), ('B','yellow'), ('C','green'), ('D','blue')]`

Part III. If the blank was `dict1.items()`, what will be the final value of `lst1`?

(a) `[]`

(b) `['A', 'B', 'C', 'D']`

(c) `['A', 'red', 'B', 'yellow', 'C', 'green', 'D', 'blue']`

(d) `['red', 'yellow', 'green', 'blue']`

(e) `[('A','red'), ('B','yellow'), ('C','green'), ('D','blue')]`

7. (1.2 points) Below are common methods and functions on strings and lists. Write the output of each cell in the box under it. Don't forget to put quotes around strings!

```
'Hamlet!'.upper()
```

```
'COGS 18'.lower()
```

```
'What it do?'.find('do')
```

```
lst = [4, 2, 1, 3]
lst.reverse()
lst.sort()
lst
```

```
lst = [4, 2, 1, 3]
lst.sort()
lst.reverse()
lst
```

```
lst = [4, 2, 1, 3]
xs = sorted(lst)
lst.remove(1)
xs
```

8. (1.1 points) The class below describes a programmer. A programmer has a name, a kind (`'novice'` or `'professional'`), years of experience, and a task.

```
class Programmer:
    def __init__(self, name, kind, years_of_experience=0):
        self.name = name
        self.kind = kind
        self.task = 'Doing nothing...'
        self.years_of_experience = years_of_experience

    def set_task(self, task):
        self.task = task
```

Part I. Complete the code below to create an instance of a `Programmer` with the name `'Brian'`, who's kind is a `'novice'`, he has `2` years of experience.

```
brian = ______________________________________________
```

Now, set `brian`'s task to `'Learning Python'`. (Use the method defined in the class.)

Part II. Write the following method, to be added to the Programmer class, which prints the programmer's name and task as: Hi! I am <name> and I am <task>. For Brian, it would print `'Hi! I am Brian and I am Learning Python'`.

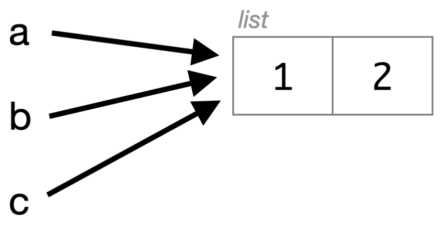
```
    def greet(self):

        print(__________________________________________________________)
```

Part III. What is the output of the following code?

```
alice = Programmer('Alice', 'professional')
print(alice.years_of_experience)
```

9. (0.6 points) Here's some code and a diagram of the resulting system state.

```
a = [1, 2]
b = a
c = b
```



Now, the following lines of code are run.

```
a[0] = None
b = [3, 4]
c[1] = None
```

Part I. What is the final value of a?

(a) `[1, 2]`

(b) `[1, None]`

(c) `[None, 2]`

(d) `[3, 4]`

(e) `[3, None]`

(f) `[None, 4]`

(g) `[None, None]`

Part II. What is the final value of b?

(a) `[1, 2]`

(b) `[1, None]`

(c) `[None, 2]`

(d) `[3, 4]`

(e) `[3, None]`

(f) `[None, 4]`

(g) `[None, None]`

Part III. What is the final value of `c`?

(a) `[1, 2]`

(b) `[1, None]`

(c) `[None, 2]`

(d) `[3, 4]`

(e) `[3, None]`

(f) `[None, 4]`

(g) `[None, None]`

Congratulations, you have reached the end! Check your answers. Did you put your UCSD email at the top of each page? You may turn in your exam and quietly exit when finished. Whew!