Practice Midterm 1

**Disclaimer: The SLs have not looked at the actual midterm when writing these questions. Let this practice midterm in no way represent what the actual midterm will be like. In addition to this practice midterm, please study past section handouts, Long Problem Assignments, and even review the lecture slides.**

1. What does it mean for something to be immutable or mutable? Additionally, name a couple of mutable types that cannot be a key in a dictionary.

2. What is an invariant and how does it help you with your programs?

Fill in the blanks for a Class definition:

3. A class describes the and of a set of objects.

4. For a class, what method must you always have (give the exact name, spelling counts)?

5. What parameter must you always have for your methods in a class and why?

6. Match the Python type to the properties of that type.

| | |
|---|---|
| List | a. Only has two valid values, used to control branches |
| Dictionary | b. Contains any real number, including fractions |
| String | c. Data structure that has no duplicates |
| Integer | d. Contains multiple items, cannot be modified |
| Float | e. Maps immutable items to elements |
| Tuple | f. Contains multiple items, can have items added |
| Set | g. Consists of 0 or more characters, immutable |
| Boolean | h. Contains discrete numbers, no fractions |

Answer the following questions given the Python code below.

```
x = {"abc": 123, "pqr": "def", "hij": (456, "abc")}
y = [123, 456, "hij", "abc", "456"]
z = ((123, "abc"), ("123abc"), ([33, 44, 55, 456]), (4,1,3,0,2))
s = "abc123pqr456"
```

7. What does each statement resolve too? (In other words, is the expression `True` or `False`?)

```
A) x[y[3]] == z[2][3]
B) x[s[0:3]] == y[0]
C) z[2][3] == x[y[z[3][4]]][0]
D) x[s[-6:len(z[2])+len(z[3])]] == x[y[2]][1]
```

8. What are the contents of x, after the following statement?

```
x[z[1][:3]] = y[len(x) + len(z[0]) - z[3][2]]
```

9. Write a function `check_palindrome(L)` where `L` is a list of strings wherein this function will return `True` if all the strings in L are palindromes or `False` otherwise. Additionally, write assert statements to ensure that all elements of L are strings and that all strings only have alphabetical characters. You may assume case insensitive.
A palindrome is a string, when reversed is still itself. The first example demonstrates that all those strings are palindromes.
Example:

```
>>check_palindrome(["Jalinnilaj", "racecar", "BoB"])
True
>>check_palindrome(["Hello", "racecar", "bob"])
False
>>check_palindrome(["Hello", 1, "bob"])
AssertionError
>>check_palindrome(["Hello", "1racecar1", "bob"])
AssertionError
```

10. Write a function `sumOfMins(grid)` that takes in a grid (2D List) of integers and returns the sum of the minimums of each column.

11. Write a function `move_to_end(phrase, chars)` where `phrase` and `chars` are strings. This function should return a string where all characters in phrase that also exist in chars are moved to the end of phrase **in the order they appear in** phrase, case-insensitive. Your final output must be uppercase.

Example:

```
>> move_to_end('hello, world!', 'h')
'ELLO, WORLD!H'
>> move_to_end('hello, world!', 'wlr')
'HEO, OD!LLWRL'
>> move_to_end('hello, world!', '!hello')
', WRDHELLOOL!'
```

12. Read in a csv file called “students.csv”. Where each line of the file is a student, and any number of courses, followed by their grades. Organize all this data into a dictionary as seen below. If a name already exists in your dictionary, add the grades, and courses of that line to the existing name in the dictionary.

Example:

```
“John, CSc 110, A, CSc 120, C”
“Kevin, CSc 110, A”
“Tony, CSc 110, B, CSc 120, C, CSc 210, C, CSc 245, B”
“John, CSc 210, C, CSc 245, A”
“Kevin, CSc 120, B”
```

Your dictionary should look like this:

```
{“John”: {CSc 110: A, CSc 120: C, CSc 210: C, CSc 245: A}
“Kevin”: {CSc 110: A, CSc 120: B}
“Tony”: {CSc 110: B, CSc 120: C, CSc 210: C, CSc 245: B}}
```

13. Write a function `movieTimeLine()` that takes in two inputs, a filename and an actor's name. Read in the file using the filename in order to create a 2D dictionary where the top level is the actors' name and the inner level is a dictionary that maps a year to the movie they were in. After the 2D dictionary is created, print out a timeline for actor using `"{}, {}".format(year, movie)` ordered by the year.

Example:

```
American Graffiti,Harrison Ford,1973
Raiders of the Lost Ark,Harrison Ford,1981
Jaws,Richard Dreyfuss,1975
The Fugitive,Harrison Ford,1993
```

Example output for Harrison Ford:

```
1973, American Graffiti
1981, Raiders of the Lost Ark
1993, The Fugitive
```

14. Given the following code. What is the output?

```
def main():
    sum1 = 1
    L = [0,2,4]
    sum2 = add_one(L)
    print(L)
    print(sum2)
    for i in range(len(L)):
        sum1 += L[i]
    print(sum1)

def add_one(lis):
    sum1 = "0"
    for i in range(len(lis)):
        lis[i] += 1
        sum1 += "1"
    return sum1
main()
```

15. For the following program, which line will cause an error, and why?

```
1.   Dict = {}
2.   Dict[5] = "abc"
3.   Dict["abc"] = 5
4.   Dict[5] = 2
5.   Dict[(1,2,3)] = 5
6.   Dict[""] = 2
7.   Dict[[18, 5]] = 12
8.   print(Dict)
```

16. Write a Python program that:
    1. Prompts the user for two numbers, N and M
    2. Creates an N by M two dimensional list, where each element of the list is the product of the row index and column index of that element
    3. And prints out the odd elements of that two dimensional list which are divisible by 3 or 5, without duplicates, in sorted order.

The output should be similar to the following:

```
n: 4
m: 9
3
5
9
15
21
```

17. Help! The Avengers are off fighting Thanos, so Tony Stark doesn't have time to check the invariants on Shield's newest application to track geolocations of current super heroes.

This program reads in a csv file containing multiple lines, with a single line holding a person's name, super power (if any), and their geolocation (for simplicity this will be a single integer). You can assume that there will be no duplicate names.

Example:

```
Stephen Strange, Mystic Arts, 323244
Mr Man, Scandalous Good Looks, 44035
Peter Parker, Anything a Spider can do, 454354
Hank Pym, Intellect, 098884
Civilian Bob, None, 456534
```

Describe what the invariant is supposed to be at Point 1 and Point 2.

```
def important_avenger_application():
     file=open("civilian_data.csv")
     superhero_dict={}

     for line in file:
          #Point 1
          person=line.strip().split(",")
          power=person[1].tolower()

          if(power not in superhero_dict and power!="none"):
               superhero_dict[power]= [(person[0],person[2])]
          superhero_dict[power].append((person[0], person[2])

     For key in superhero_dict:
          superhero_dict[key].sort()
     #Point 2
     file.close()
```

18. Billy Bob Jones down the street has just started a lemonade stand and wants and easy way to track the amount and types of lemonade he has sold. Write a `class LemonadeStand` that has the following requirements.

Attributes:
`lemonade_dict`: dictionary of lemonades sold that day

Methods:
`__init__(self)`: Initializes lemonade_dict
`__str__(self)`: Produces lemonade and the amount sold, one per line (sorted order)
`get_amount(lemonade_string)`: Returns the amount sold of the `lemonade_string` as an integer
`add_lemonade(lemonade_string)`: Adds `lemonade_string` to possible lemonades that can be sold
`start_new_day()`: clears out the currently stored lemonades to start a new day

Example Output:

```
>>>new_stand=LemonadeStand()
>>>print(new_stand)

>>>new_stand.add_lemonade(“Red”)
>>>new_stand.add_lemonade(“Blue”)
>>>new__stand.add_lemonade(“Red”)
>>>print(new_stand)
Red: 2
Blue: 1
>>>new_stand.start_new_day()
>>>print(new_stand)

>>>
```

*Page left blank for answer.*

19. Currently, you are writing a `class Singer` in order to sort the scores of each singer in a contest. A `Singer` object is composed of three scores, one from each judge and the name of the singer.

Write the special function to allow for this capability of being able to sort singers given the following `sort_singers(lis)` function. `lis` is a list of `Singer` objects.

```
class Singer:
  def __init__(self, name, aScore, bScore, cScore):
      #Scores from each judge
       self._name = name
          self._aScore = aScore
          self._bScore = bScore
          self._cScore = cScore

    def __str__(self):
          return self._name

<<SPECIAL FUNCTION GOES HERE>>
```

```
def sort_singers(lis): # What sorting algorithm is this?
    for i in range(len(lis)-1, 0, -1):
        for j in range(i):
            if lis[j] >= lis[j+1]: # Comparing Singer objects
                temp = lis[j]
                lis[j] = lis[j+1]
                lis[j+1] = temp
```

20. Write a class for a car that takes make, model, and top speed. Then write a method to calculate the time it would take for the car to travel a certain distance and a method that prints out the attributes of the class

Example:

```
>> dreamcar = Car("Lamborghini","Aventador",216)#216 in miles
per hour
>> print(dreamcar)
Lamborghini Aventador has the top speed of 216 mph
>> dreamcar.shortest_time(432)#in miles
2
```

21. Write a class that keeps track of all the lightsabers in General Grevious's collection. With the following methods:

- A method that adds lightsabers to his collection that should take in as a parameter the color of the lightsaber and the previous owner.
- A method that returns the amount of each color lightsaber in the collection.
- A print method that prints out the name of the previous owners lightsabers in the collection.

Example:

```
>> New_collection = Collection()
>> New_collection.add("Anakin Skywalker","blue")
this will make a fine addition to my collection
>>
>> New_collection.add("Obi-Wan Kenobi","Blue")
this will make a fine addition to my collection
>>
>> New_collection.colors_in_collection()
2 blue
>> print(New_collection)
Anakin Skywalker
Obi-Wan Kenobi
```

*Page left blank for answer.*

22. You have been hired by Twitter to help write part of a new and improved algorithm to rank trending hashtags! Your job is to write a function that parses raw tweets and outputs useful data on the hashtags contained in these tweets.

Write a function `parse_tweets(tweets)` where `tweets` are data on all tweets posted in the last hour, represented as a list of tuples in the form of `(user,tweet)`. This function should return a multi-level dictionary containing data about the hashtags in the form as specified in the example below.

The hashtag may be located anywhere in the tweet, but you are guaranteed that it will be separated by whitespace and will contain no punctuation characters.
Example:
>> tweets = [('user1', 'I <3 OOP! #csc120'), ('user2', '@user1 me too! #csc120'), ('user3', 'Can someone help me with my #csc110 homework?'), ('user1', 'Go to the 228 lab for help in #csc120')]
>> parse_tweets(tweets)

```
{
   '#csc120': {
      'tweets': {
         'user1': ['I <3 OOP! #csc120','Go to the 228 lab for help in #csc120'],
         'user2': ['@user1 me too! #csc120'],
      },
      'count': 3,
      'unique_users': 2
   },
   '#csc110': {
      'tweets': {
         'user3': ['Can someone help me with my #csc110 homework?']
      },
      'count': 1,
      'unique_users': 1
   }
}
```

*Page left blank for answer.*

```
import math

def get_prime_factors(num):
    primes_list = []
    while num % 2 == 0:
        primes_list.append(2)
        num = num // 2

    #POINT 1
    for i in range(3, int(math.sqrt(num)) + 1, 2):
        while num % i == 0:
            primes_list.append(i)
            num = num // i

    if num > 2:
        primes_list.append(num)

    #POINT 2
    return primes_list
```

23. Given this function, what are the invariants (in English) at Points 1 and 2?

What would be the assertion to check at Point 1 instead of the comment?

24. Given this function, what are the invariants (in English) at Points 1 and 2?

```
def mystery_func(str):
     for i in range(0, len(str)):
          #POINT 1
          if (str[i] == str[len(str) - 1 - i])
               continue
          return False

     #POINT 2
     return True
```