CSCSCI-UA.0002 – Final Exam Practice Questions Set 2

There is a greater number of *difficult* questions in the practice set, and some of the topics may be extra credit.

1.	At the end of this program, what values do the varight of the code. The order of the key value p	riables d1, d2, a and s contain? Write your answers in the space provided to the pairs will not be taken into account. (2 points)					
	<pre>d1 = {'first_name':'Stanley', 'year_ d2 = {'first_name':'Ridley', 'year_b</pre>	born':1928}					
	<pre>a = dl.get('last_name', 'Unknown') d2['last_name'] = 'Scott' s = ''</pre>	OIR -1237/					
	<pre>for k, v in d2.items(): s += str(v)</pre>						
	d1						
	d2						
	a						
	s						
2.	Find 2 values that a user can input that will cause blanks below the code. (2 points)	the following program to crash. Why is an error caused by the input? Fill in the					
	import random						
	<pre>cards = ['empty', 'empty', 'penny'] random.shuffle(cards)</pre>						
	number = int(a)	<pre>a = input('Please choose a card: 0, 1 or 2\n>') number = int(a)</pre>					
	<pre>if cards[number] == 'penny': print('you guessed right!')</pre>						
	<pre>else: print('nope, nothing there')</pre>						
	a) Error Causing Input 1:	Reason for Error:					
	b) Error Causing Input 2:	Reason for Error:					
3.	errors from user input. The error messages that	exception handling so that a different message is displayed for each of the possible are printed out can be of your choosing (for example, "bad input" for one and e from the previous program in the space below. (3 points)					
4.	What is the output of the following program? Use output can be placed in a single box. Leave the boxes. (3 points)	e the grid to the right of the program as a guide; each individual character of e a box blank to represent a space character. You do not have to use all of					
	<pre>def create_table(size, letters): table = ''</pre>						
	for i in range(1, size + 1): row = ''						
	for c in letters: row += str(i) + c + ' '						
	table += row + '\n'						
	return table						
	<pre>def main(): num, s = 4, 'abc'</pre>						
	<pre>print(create_table(num, s)) main()</pre>						

- $5. \quad Complete \ the \ two \ function \ definitions \ below \ for \ \verb|square_the_odds| \ and \ \verb|square_the_odds_in_place|.$
 - a) It should take a list of integers as a parameter
 - b) It should either:
 - return a **new** list with all of the elements in the original list, except with every odd number squared...
 - modify the list passed in *in place* (the original list passed in will be modified), so that the odd numbers are squared...
 - c) For example: $[1, 2, 3, 4, 5, 6, 7]^{(1)}$ [1, 2, 9, 4, 25, 6, 49] (either returned new or in place)
 - d) Don't worry about input that's not a list of integers.

```
def square_the_odds(numbers):

def square_the_odds_in_place(numbers):
```

6. Write the output of the following code in the space to the right.

```
numbers = [1, 2, 3]
same_numbers = numbers
copy_numbers = numbers[:]
print('part 1')
print(numbers)
print(same_numbers)
print(copy_numbers)
result1 = same_numbers.append(4)
result2 = copy_numbers.pop()
print('part 2')
print(result1)
print(result2)
print('part 3')
print(numbers)
print(same_numbers)
print(copy_numbers)
copy_numbers.append(numbers)
copy_numbers.extend(same_numbers)
print('part 4')
print(copy_numbers)
```

- 7. Create a function called fib that generates a Fibonacci sequence. An example sequence is 0, 1, 1, 2, 3, 5, 8, 13, 21, 34. By definition, the first two numbers in the Fibonacci sequence are 0 and 1, and each subsequent number is the sum of the previous two.
 - a) it should take a single parameter, the number of digits to generate (remember, the first two are always 0 and 1)
 - b) it will not return anything; instead, it should print out each digit on a new line
 - c) for example: fib(10) 0, 1, 1, 2, 3, 5, 8, 13, 21, 34
 - d) hint: again, the sequence starts with 0 and 1, so the numb that need to be generated programmatically are actually 2 less than the argument passed in
 - e) hint: swapping may help

hint: you'll have to keep track of the previous 2 and the calculated value

- 8. Create a function called encrypt. It will create a new string from an input string, with certain characters substituted. (3 points)
 - a) it should take 2 arguments, a dictionary that has characters as keys and values as translated characters
 - b) the encrypted string will be the original string inputted, and the translated string will have the letters a, b, c, d, and e substituted with punctuation marks, and with everything else substituted with dashes ('-')
 - c) the 'translation guide' for the letters will be stored in a dictionary that is passed on into the function call
 - d) for example:

```
\# assume that the dictionary d exists, and you can use it pass in to the function d = {'a':'!', 'b':'@', 'c':'\#', 'd':'\$', 'e':'\%'} print(encrypt(d, 'cab'))  #!@ print(encrypt(d, 'modes'))  --\$\*-
```

e) the 'everything else' substituted by a dash character is not specified in the dictionary; that particular logic can be handled by your program

a)	Name two operations and/or functions that are supported by sequence types (lists, strings, and tu		
	(1)(2)		
b)	o) Give two examples of how lists and strings different (excluding different methods)		
	(1)		
	(2)		

c) What's the difference between a tuple and a list?

9. Answer the following questions about lists, strings and tuples. (3 points)

10. You love pizza parties, but organizing them is a *drag*... especially finding out how many pies to purchase! Write a short program that helps you calculate the number of pies to purchase for a pizza party.

Part 1: Define a function called how_many_pies. It should take two arguments: the number of people eating pizza and the number of slices each person will have (this will be the same for every person; none of that 3 for her and 2 for him stuff!).

- 1. a) It should assume that every pie comes with 8 slices.
- 2. b) The function will always over-order pies... meaning that if the number of pies has to be rounded, round up.
- 3. c) For example, if 5 people are coming to the party, each person wants 2 slices, and each pie has 8 slices, we would want 2 pies

to accommodate everybody!

- 4. d) You don't have to worry about non-integer, zero or negative input
- 5. e) You can use a function in the math module called ceil to round up. It takes one argument and it returns the smallest integer

value that's greater than or equal to the original argument. For example: $\mathtt{math.ceil}(1.2)\ 2$.

6. f) An example run of the function itself: print(how_many_pies(9, 3)) 4

Part 2: Use this function after asking from input for the user. (That is, assume that you've already written your function in the same file; you just have to use it below). It should ask for number of people and number of slices. You don't have to worry about non-integer input. Example interaction below (everything after > is user input):

```
How many people?
>9
How many slices?
>3
You'll need 4 pies for a pizza party
```

11. True or False (3 points)

12. List three methods that you can call on file objects, along with what they do:

13. What data structure (data types or combination of data types) would you use to hold a word and all of its synonyms. The data structure should be flexible enough to add and/or remove synonyms. For example, lists, strings, tuples, some combination of data types (a dictionary of tuples), etc.?

```
good - bully, cracking, great
```

14. Imagine that the code in the first column is executed line-by-line. After each line is executed, examine the state of the variables a, b, and c, and write the values that are bound to them in the columns to the right of the code. The variables may contain a value, may not yet exist or may have a value that hasn't changed yet (that is, it's the same as the previous value that it had).

For example, after the code in the first row is executed, a contains [21, 7, 14], while b and c do not exist yet. Some of the table is already filled in; fill in the remainder of the missing values. (3 points)

Code	а	b	С
a = [21, 7, 14]	[21, 7, 14]	does not exist	does not exist
b = [35, 35, 35]	[21, 7, 14]	[35, 35, 35]	does not exist
a.append([14, 28])		[35, 35, 35]	does not exist
b.extend(a.pop())			does not exist
c = a.sort()			

15. Write a **recursive** version of a factorial function: (4 points)

fact(4) ^ 24