COSC 101, Exam #2
November, 2020

Name: ______________________________

Please write your name above. Do not start the exam until instructed to do so.

You have (2 * 1.5) = 3 hours to complete this exam.

There are 13 questions and a total of 61 points available for this exam. Don't spend too much time on any one question.

Since indentation is important in Python, please be sure that your use of indentation is obvious for any code you write.

If you want partial credit, show as much of your work and thought process as possible.

If you run out of space for answering a question, you can continue your answer on one of the blank pages at the end of the exam. If you do so, be sure to indicate this in two places: (1) below the question, indicate which blank page contains your answer, and (2) on the blank page, indicate which question you are answering.

Question	Points	Score
1	1	
2	4	
3	4	
4	6	
5	2	
6	2	
7	4	
8	9	
9	5	
10	6	
11	6	
12	2	
13	10	
Total:	61	

1. (1 point) This take home exam is held to the same academic honor code standards as an in person exam. While working on this exam you are permitted to reference your textbook, notes, and material on the course Moodle. You are not permitted to use Python, other outside sources (such as Google searches), nor communicate with others.

   You may use scratch paper for writing and working through problems.

   To that show you understand the expectations about taking this exam, circle the statements below that are allowable under the honor code:

   A. Search the internet for solutions or hints

   B. Run Python code in IDLE, Repl.it, the textbook website, or using any other development environment

   C. Post exam questions or answers online

   D. Talk to a classmate about this exam

   E. Look at my notes

   F. Read the online textbook

   G. Look at the course Moodle page

2. (4 points) What is printed by the following program?

```python
a = "bright"
c = "!"
def function1(a, b):
    c = a[:1] + b
    print("C:", c)
    if b in a:
        return "a"
    return c

def function2(b):
    a = b[1:]
    print("A:", a)
    print("B:", b)
    return function1(b, a)

function2(a)
print("C:", c)
```

3. (4 points) What is printed by the following program?

```python
lst = ['br', -5]
def function3(lst):
    for i in range(len(lst)):
        lst[i] *= 2
    lst.append('!')
    print(">", lst)

print(">", lst)
lst = function3(lst)
print(">", lst)
```

4. (6 points) A genie appears and presents you with an option of two gifts:

   1. A regular bag with \$100,000
   2. A magic bag containing 1 penny that magically doubles the amount of money inside every day

   You want to know how many days it would take before the money in the magic bag (option 2) is more than the money in the regular bag (option 1).

   Write a Python script in the space below that answers this question:

5. (2 points) Which of the following are valid nested list definitions?

   Circle one or more

   A. `lst = [['Colgate'] ['University']]`

   B. `lst = [[]]`

   C. `lst = ['Colgate', 'University']`

   D. `lst = [['Colgate', 'University']`

   E. `lst = [[1, 13], [2, 13346], [3, 2020]]`

   F. `lst = [['Oak', 'Drive'], ['Hamilton'], ['New', 'York']]`

6. (2 points) Assume lst is a non-empty list of lists, where the nested lists are non-empty and contain integers. Which of the following are valid expressions (i.e., expressions that do not lead to a syntax or runtime error)?

   Circle one or more

   A. `sum(lst[-len(lst)])`

   B. `lst[len(lst)]`

   C. `lst[0][:len(lst[0])]`

   D. `lst[-1].sort()`

   E. `lst[0][0]`

7. (4 points) Assume that the variable board contains a representation of a 3x3 tic-tac-toe board, where board is length 3, and each nested list is of length 3. Assume also that an entry in each nested list contains ‘x’, ‘o’, or ‘’ (empty string), depending whether an ‘x’ occupies a position, ’o’ occupies a position, or a position is empty.

   For example, the board

   o | o |
   - + - + -
   x | o | x
   - + - + -
   x | |

   would be represented by the nested list [[‘o’,‘o’,‘’], [‘x’,‘o’,‘x’], [‘x’,‘’,‘’]]

   Consider the following check_diagonal(board, player) functions, which accept a tic-tac-toe board in this format and a player (‘x’ or ‘o’). Which of these functions correctly checks whether a given player has won the game along the top-left to bottom-right diagonal?

   Circle one or more

   A.

```
def check_diagonal(board, player):
    for i in range(len(board)):
        for j in range(len(board)):
          if i == j and board[i][j] != player:
              return False
    return True
```

   B.

```
def check_diagonal(board, player):
    for i in range(len(board)):
        if board[i][i] == player:
            return True
    return False
```

   C.

```
def check_diagonal(board, player):
    for i in range(len(board)):
        for j in range(len(board)):
            if board[i][j] != player:
                return False
    return True
```

   D.

```
def check_diagonal(board, player):
    s = ''
    for i in range(len(board)):
        for j in range(len(board)):
            if i == j:
                s += board[i][j]
    return s == player*3
```

   E.

```
def check_diagonal(board, player):
    for i in range(len(board)):
        if board[i][i] != player:
            return False
    return True
```

8. (9 points) Complete this program by filling in the boxes with the appropriate list methods and following the instructions in the comments.

```
# For example, if you were asked to convert y to uppercase,
# you would write upper() into the box below
y = y.

# Take listX and split it into two lists (listA and listB), where
# listA contains only numbers and listB contains only strings.
listX = ["i", "found", "a", "1000000", "dollars", "in", "a",
         "5", "and", "10", "cent", "store"]
listA = []
listB = []

for item in listX:
    if item.isnumeric():
        num = int(item)
        listA.

    elif item.isalpha():
        listB.

# put listA in numerical order from lowest to highest
listA.

# change listA a from this: [5, 10, 1000000]
# to this: [5, 10, 100, 1000000]
listA.

# remove and print the last element in listB


# put listB in reverse alphabetical order
listB.

listB.

# print the number of times 'a' appears in listB


print(listA) # [5, 10, 100, 1000000]
print(listB) # ['in','i','found','dollars','cent','and','a','a']
```

9. (5 points) Write a function called piratify that has one string parameter s. The function should create a new string identical to s but with all "ar" substrings (case-INsensitive) replaced with "ARRR". The function should return this new string in all uppercase letters.

   For example:

   - piratify ("Hard to starboard!") should return "HARRRD TO STARRRBOARRRD!"
   - piratify ("Aren't ye a salty dog?") should return "ARRREN'T YE A SALTY DOG?↘ "
   - piratify ("Raise the mizzen") should return "RAISE THE MIZZEN"

   Rearrange the following lines of code to write the function. Each line may only be used once, and some lines may not be used at all.

   - `s = "ARRR".join(s)`
   - `s = s.upper()`
   - `s[i] = "ARR"`
   - `s = s.split("r")`
   - `return s`
   - `def piratify(s):`
   - `s = s.split("AR")`
   - `print(s)`
   - `i = s.find("AR")`
   - `s = s[0] + "ARRR" + s[1]`
   - `s = s.lower()`

10. (6 points) Write a function called bop_it that has one integer parameter n. The function should create a string with n repetitions of the form [verb] it! separated by spaces, where [verb] is one word selected uniformly at random from the list ["Bop", "Twist", "Pull"]. If n is 0 or negative, the function should return the empty string. For example:

    - bop_it(3) COULD return "Bop it! Twist it! Bop it!" (depending on the random choice)
    - bop_it(2) COULD return "Pull it! Bop it!" (depending on the random choice)
    - bop_it(0) should return ""

    Rearrange the following lines of code to write the function. Each line may only be used once, and some lines may not be used at all.

    - `choice = verbs[random.randint(0,3)]`
    - `verbs = ["Bop", "Twist", "Pull"]`
    - `result = choice * n`
    - `choice = verbs[random.randint(0,2)]`
    - `return result.strip()`
    - `def bop_it(n):`
    - `result += choice + "it! "`
    - `verbs = "Bop Twist Pull"`
    - `choice = verbs.split(" ")`
    - `return result`
    - `for i in range(n):`
    - `result = ""`
    - `for i in range(len(verbs)):`

```
import random
```

11. (6 points) Write a function called is_sorted that has one string parameter. The parameter is assumed to be a string with multiple words separated by spaces. The function should return True if all of the words in the string are in case-INsensitive alphabetical order and False otherwise.

    For example:

    - is_sorted('Apple kiwi mushroom Pear') should return True
    - is_sorted('Apple Aardvark Alien') should return False
    - is_sorted('zoom zoom zoom') should return True

    Rearrange the following lines of code to write the function. Each line may only be used once, and some lines may not be used at all.

    - `for i in range(len(words)):`
    - `if words[i] > words[i+1]:`
    - `for i in range(len(words) - 1):`
    - `return False`
    - `if words[i] <= words[i+1]:`
    - `if words[i].upper() <= words[i+1].upper():`
    - `def is_sorted(s):`
    - `return True`
    - `if words[i].lower() > words[i+1].lower():`
    - `words = s.split(" ")`

12. (2 points) Assume that the file protagonists.txt has the following contents:

```
Odysseus, The Odyssey
Liu Bei, Romance of the Three Kingdoms
Arjuna, Mahabharata
```

If the following program is run from the same directory as protagonists.txt what are the final contents of protagonists.txt?

Circle one

A.

```
Odysseus, The Odyssey
Liu Bei, Romance of the Three Kingdoms
Arjuna, Mahabharata
Beowulf, Beowulf
Prospero, The Tempest
```

B.

```
Beowulf, Beowulf
Prospero, The Tempest
```

C.

```
Odysseus, The Odyssey
Liu Bei, Romance of the Three Kingdoms
Arjuna, Mahabharata
Beowulf, BeowulfProspero, The Tempest
```

D.

```
Beowulf, BeowulfProspero, The Tempest
```

E.

```
Prospero, The Tempest
```

13. (10 points) Assume that you have a comma-separated values (CSV) file named cereal.csv with nutrition information about popular cereals. Each line of the file describes one particular cereal. For example, the first 5 lines of the file are as follows:

```
Name,Manufacturer,Type,Calories,Protein,Fat,Sugars
Cap-n-Crunch,Quaker Oats,Cold,120,1,2,12
Cheerios,General Mills,Cold,110,6,2,1
Cinnamon Toast Crunch,General Mills,Cold,120,1,3,9
Cream of Wheat,Nabisco,Hot,100,3,0,0
```

The values for Protein, Fat, and Sugars are reported in grams. Note that the first (header) line of the file is different from the rest of the lines and will need to be handled differently.

Write a function most_sugary() that reads this file and finds the COLD cereal with the highest ratio of sugars to calories. The function should write the name of this cereal along with its sugars/calories ratio as a single line in a new file, most_sugary.txt. This line should have the format [cereal name] has the highest sugars/calories ratio of [ratio].

(This page is intentionally blank. Label any work with the corresponding problem number.)

(This page is intentionally blank. Label any work with the corresponding problem number.)