COSC 101, Exam #3
April 2022

Name: _____

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

You have 50 minutes to complete this exam.

There are **??** questions and a total of **??** 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.

The last page of the exam contains documentation for string and list methods.

Run LaTeX again to produce the table

1. (6 points) Assume that the following statements have already been executed:

```
def fruit(x):
        x.append('apple')
        y = x[:]
        y.append('pear')

a = ['lime']
b = ['orange']
```

Evaluate the following expressions **in order** and write the resulting values of **both** variables `a` and `b` after each statement executes. Optionally, also draw the memory diagrams for partial credit.

(a) `b = fruit(a)`

(b) `b = a`

(c) `a[0] = 'lemon'`

2. (12 points) Write a function called `str2list` that takes in a string and returns a list. The function needs to split the string using the underscore (_) delimiter. Each element in the list needs to also have the first letter capitalized.

   Here's the doctests for the required function:

```
def str2list(txt):
    """
    >>> str2list("ab_cd_ef")
    ['Ab', 'Cd', 'Ef']
    >>> str2list("job ana_bill phil")
    ['Job ana', 'Bill phil']
    >>> str2list("1job_ana_bill_phil")
    ['1job', 'Ana', 'Bill', 'Phil']
    """
```

3. (15 points) The following code contains some errors. If a line contains an error, cross out the line and write the correct code next to it. If a line contains no errors, write nothing. Assume all comments are correct.

   For example, if you saw the following line:

   test = hello world

   You would cross it out and write the correct code like so:

   ~~test = hello world~~ test = "hello world"

   Read the docstring of the function carefully!

```
def funk(x, y):
  """ (list, str) -> list
  >>> funk(['abc', 'def', 'ghi'], 'f')
  ['ghi', 'ABC', 'de!', 'e', 'ghi', 'ABC']
  """

  # make a copy of x that we can change
  # without modifying the original list
  c = x

  # Loop over each element in the list copy and
  # see if the parameter y occurs in the
  # element. If so, replace that element with a
  # new string in which y is replaced with '!'.
  for i in c:
    if c[i].find(y) == 1:
      c[i].replace(y, '!')

  # remove the last element of the list copy
  # and add the removed element to
  # the beginning of the list copy
  a = c[-1]
  c = c.insert(0, a)

  # convert the second element of the list copy to uppercase
  c[1].upper()

  # add the parameter y to the end of the list copy as
  # a new element
  c = c.append(y)

  # get a slice of the first two elements of the list copy c
  s = c[:3]

  # concatenate lists c and s and return the result
  return c.append(s)
```

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

4. (10 points) Suppose the file with name **hamilton.csv** contains the following:

```
Season,Warmth,Climate
winter,cold,snowy
spring,warm,rainy
summer,hot,sunny
autumn,cool,windy
```

Your goal is to write a program that would read line-by-line from **hamilton.csv** and write to **hamilton.txt** the following:

```
In Hamilton, winter is cold and snowy, spring is warm and rainy, summer is hot and sunny, autumn is cool and windy.
```

On the following page is a collection of lines of code that you must choose from to construct your program. Specifically, **you must select one line of code from each letter group in order** (i.e. first select one line of code out of A1-A2, then one line of code from B1-B2, and so on) to compose your program.

Write your solution in the lines on the following page.

A1 `fin=open("hamilton.csv")`
A2 `with open("hamilton.csv","r") as fin`

B1 `fout=open("hamilton.csv","wr")`
B2 `fout=open("hamilton.txt","w")`

C1 `fin.read()`
C2 `fin.readline()`

D1 `fout.write("In Hamilton")`
D2 `fout.print("In Hamilton")`

E1 `while line in fin:`
E2 `for line in fin.readlines():`

F1 `  data = line.split(",").strip()`
F2 `  data = line.strip().split(",")`

G1 `  fout.print(data[0],"is",data[1],"and",data[2])`
G2 `  fout.write(", " + data[0] + " is " + data[1] + " and " + data[2])`

H1 `fout.write(".")`
H2 `fout.close()`

I1 `fout.close()`
I2 `fin.close()`

J1 `fout.write(".")`
J2 `fout.close()`

Solution:

A1,B2,C2,D1,E2,F2,G2,H1,I2,J2

5. (7 points) For the following program:

   1. Circle every line of code at which an exception may occur
   2. Write the program's printed output, assuming that the user intends to enter `hello` then `0`

```
def main():
    """Main function collects values from user into a list"""
    lst = []

    try:
        value1 = int(input())
        lst.append(value1)

        value2 = value1 + 10
        lst.append(value2)

        value3 = int(input())
        lst.append(value3)

    except ValueError:
        print("Oops, that's not an integer!")
        lst.append(-1)

    for i in range(len(lst)):
        print(lst[i])

main()
```

Solution:

```
Oops, that's not an integer!
-1

value1 = int(input()) should be circled
value2 = int(input()) should be circled
```

(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.)

String methods

- `upper()` — Returns a string in all uppercase
- `lower()` — Returns a string in all lowercase
- `capitalize()` — Returns a string with first character capitalized, the rest lower
- `strip()` — Returns a string with the leading and trailing whitespace removed
- `lstrip()` — Returns a string with the leading whitespace removed
- `rstrip()` — Returns a string with the trailing whitespace removed
- `count(item)` — Returns the number of occurrences of item
- `replace(old, new)` — Replaces all occurrences of old substring with new
- `center(width)` — Returns a string centered in a field of width spaces
- `ljust(width)` — Returns a string left justified in a field of width spaces
- `rjust(width)` — Returns a string right justified in a field of width spaces
- `find(item)` — Returns the leftmost index where the substring item is found, or -1 if not found
- `rfind(item)` — Returns the rightmost index where the substring item is found, or -1 if not found
- `index(item)` — Like find except causes a runtime error if item is not found
- `rindex(item)` — Like rfind except causes a runtime error if item is not found
- `split(separator)` — Return a list of the words in the string, using separator as the delimiter string
- `join(lst)` — Return a string which is the concatenation of the strings in lst
- `isalpha()` — Return True if all characters in the string are alphabetic and there is at least one character
- `isdigit()` — Return True if all characters in the string are decimal characters and there is at least one character
- `islower()` — Return True if all cased characters in the string are lowercase and there is at least one cased character
- `isspace()` — Return True if there are only whitespace characters in the string and there is at least one character
- `isupper()` — Return True if all cased characters in the string are uppercase and there is at least one cased character

List methods

- `append(item)` — Adds a new item to the end of a list
- `insert(position, item)` — Inserts a new item at the position given
- `extend(lst)` — Extend the list by appending all the items from lst
- `pop()` — Removes and returns the last item
- `pop(position)` — Removes and returns the item at position
- `sort()` — Modifies a list to be sorted
- `reverse()` — Modifies a list to be in reverse order
- `index(item)` — Returns the position of first occurrence of item
- `count(item)` — Returns the number of occurrences of item
- `remove(item)` — Removes the first occurrence of item
- `copy()` — Return a clone of the list
- `clear()` — Remove all items from the list