CS 301 - Fall 2015

Instructor: Laura Hobbes LeGault

Midterm Exam 2 — 16.67%

Full name: ______________________________

Student ID #: ______________________________

I certify that I will keep my answers covered and do my best to not allow my exam paper to be viewed by another student during the exam or prior to completion of their exam. I also certify that I have not viewed or any way used another's work in completing my answers. I understand that being caught allowing another to view my work or being caught viewing another's work are both violations of this agreement and either will result in automatic failure of the course and an academic misconduct letter to the Deans Office for myself and any other individuals involved.

Signature: ______________________________

The following exam has 20 questions and is worth a total of 88 points. You will have 50 minutes to complete the exam. **Be sure to read through every question completely.**

The questions on the exam are as follows:

1. **Dual Choice** — 8 questions worth 2 points each.
2. **Multiple Choice** — 9 questions worth 4 points each. Choose the *best* answer.
3. **Fill-in-the-blank** — 6 blanks worth 6 points each. Be complete.

You may not use notes or books, your neighbors, or calculators or any other electronic devices on this exam. **Turn off and put away** your cell phone, pager, Inspector Gadget Watch, etc. now.

Disclaimer: the following are provided for your reference only, and the inclusion of information here does not guarantee it will be used on the exam.

Operator Precedence Table:

level	operator	description
higher	`( <expression> )`	grouping with parentheses
	`x[index:index]`	slicing
	`x[index]`	indexing
	`* / %`	multiplicative
	`+ -`	additive
↕	`< <= > >=`	relational
	`== !=`	equality
	`not`	logical not
	`and`	logical and
lower	`or`	logical or
	`= += *=`	(compound) assignment

Built-in functions:

`raw_input(p)`	Prompts the user for input using `p` and returns the user's input as a string.
`len(s)`	Return the length (the number of items) of an object.
`range(n)`	Returns a list of `n` consecutive integers beginning at `0`.
`type(x)`	Returns the *data type* of the value stored in `x`

Constants and functions from the math module:

`math.pow(x,y)`	Returns `x` raised to the power `y`. Converts both arguments to `floats`.
`math.pi`	The mathematical constant $\pi = 3.141592...$

Functions from the os module:

`os.getcwd()`	Return a string representing the current working directory.
`os.chdir(path)`	Change the current working directory to `path`.
`os.rename(old, new)`	Rename the file or directory `old` to `new`.

List and dictionary methods:

`list.append(x)`	Add the value `x` to the end of `list`.
`list.insert(i,x)`	Insert the value `x` at the `i`th index of `list`.
`dict.keys()`	Return a copy of `dict`'s list of keys.
`dict.values()`	Return a copy of `dict`'s list of values.

A or B: Terminology

1. If a function does not include an explicit return statement, it returns the value ____________. (2)

   ○ Null

   ○ None

2. List elements are accessed by ______________. (2)

   ○ index

   ○ key

3. A tuple is an example of a(n) ______________ data type. (2)

   ○ mutable

   ○ immutable

4. In a file system, a directory is a type of ______________. (2)

   ○ file

   ○ path

5. Information is passed *out of* a function via its ______________. (2)

   ○ parameters

   ○ return values

6. The Python expression (2)

   ```
   word in sentence
   ```

   is an example of a ______________.

   ○ loop header

   ○ boolean statement

7. Elements of a dictionary are ordered by ______________. (2)

   ○ random

   ○ hash value

8. When slicing an iterable using `[a:b:c]` syntax, the value at `b` is the ______________. (2)

   ○ step size

   ○ upper bound

Multiple Choice: Reading code

9. Which of these lists is produced by the following code? Assume the math module has been imported, and **caution** – make sure you read the code carefully! (4)

```
[math.pow(n,2) for n in range(5)]
```

○ `[1.0, 4.0, 9.0, 16.0, 25.0]`

○ `[0, 1, 4, 9, 16]`

○ `[0.0, 1.0, 4.0, 9.0, 16.0]`

○ `[1.0, 2.0, 4.0, 8.0, 16.0]`

10. Which of the following loops will **not** correctly create a list of the *values* in some dictionary `d`? Assume the variable `val = []` has already been created. (4)

○
```
for key in d:
      val.append( d[key] )
```

○
```
i = 0
while i < len(d.keys()):
      val.append( d[d.keys()] )
```

○ `val = list(d.values())`

○ `val = [ d[i] for i in d ]`

11. Which of the following `for` loop headers correctly translates the following `while` loop construction? (4)

```
i = len(my_list)-1
while i > 0:
   x = my_list[i]
   i -= 2
```

○ `for i in my_list[::-2]`

○ `for x in my_list[::-2]`

○ `for i in my_list[::2]`

○ `for x in my_list[::2]`

12. What is the value in `my_list` after the following code executes? Be careful – trace through the code completely! (4)

```
my_list = [1, 2, 3, 4]

def shift():
   temp = my_list[0]
   for i in range( len(my_list) - 1 ):
      my_list[i] = my_list[i + 1]

shift()
```

○ `[2, 3, 4, 1]`

○ `[4, 1, 2, 3]`

○ `[2, 3, 4, 4]`

○ The code produces a `NameError`.

13. Given the list `phone` below, which of the following statements will put the value `'t'` into the variable `letter`? (4)

```
phone = [ ['abc', 'def'],
          ['ghi', 'jkl', 'mno'],
          ['pqrs', 'tuv', 'wxyz'] ]
```

○ `letter[0] = phone[2][1]`

○ `letter = phone[0][2[1]]`

○ `letter = phone[2][1][0]`

○ `phone[2[1[0]]] = letter`

14. Given the following valid combination of function calls, what *type* of value is passed to the function `my_fcn()`? (4)

```
my_fcn( len( str( range(15)[:3] )))
```

○ `int`

○ `list`

○ `str`

○ `bool`

15. Given a matrix of ints `matrix = [ [1,2,3], [4,5,6], [7,8,9] ]` which of the following replacements for `CODE` correctly calls the `swap()` function to transform `matrix` to its *transpose*? (4)

```
def swap( list1, ind1, list2, ind2 ):
   """ Swaps list1[ind1] with list2[ind2]; assume correctly implemented """

for i in range(len(matrix)):
   for j in range(i, len(matrix[i])):
      CODE
```

Recall: $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$.

○ `CODE: swap(matrix, j, matrix)`

○ `CODE: swap(matrix, i, matrix, j)`

○ `CODE: swap(matrix[i], j, matrix[j], i)`

○ `CODE: swap(matrix[i], i, matrix[j], j)`

16. Now let's look at that swap function – which of the following replacements for `VALUE1` and `VALUE2` correctly swap the values in the lists? (4)

```
def swap( list1, ind1, list2, ind2 ):
   """ Swaps list1[ind1] with list2[ind2] """
   temp = VALUE1
   list1[ind1] = VALUE2
   list2[ind2] = temp
```

○ `VALUE1:  ind1         VALUE2:  list2[ind2]`

○ `VALUE1:  ind1         VALUE2:  ind2`

○ `VALUE1:  list1[ind1]  VALUE2:  list2[ind2]`

○ `VALUE1:  list1[ind1]  VALUE2:  ind2`

17. Which of the following types is **not** a legal dictionary *value*? (4)

○ `str`

○ `int`

○ `dict`

○ All of the above are legal dictionary value types.

Fill-in-the-blank: Writing code

Fill in the blanks to make the functions behave as the comments indicate. Each line is worth **6 points**, and there are a total of 6 lines.

18.
```
def ________________________                                                    (6)
     """ Given the function call below, finish this function """

      ________________________ len(str(x))                                     (6)
  print (my_function(14))    # prints ONLY the number 2
```

19. Remember that each element in matrix multiplication is given by $AB_{ij} = \sum_k A_{ik}B_{kj}$.
```
  def multiply(m1, m2):
       """ Prints the result of multiplying matrix m1 by matrix m2 """
       for i in range(len(m1)):
           for j in range(len(m1[i])):
               element = 0
               for k in range(len(m1[i])):

                  ____________________________    # add A(i,k)×B(k,j) to sum   (6)

               ____________________________    # display the element             (6)
           print()    # prints a newline after each row
```

20.
```
  import os
  def move(filename, dir):
       """ Move the file named filename in the current working directory
           into the directory specified by the path in the variable dir
           (See below for an example call to this function.)
       """
       newpath = ______________________________________                        (6)

       os.rename( ___________________, ___________________ )                    (6)

  move(‘e2.txt’, ‘Exams/CS301’)    # example call to move() function
```