CS115 Fall 2014 Test 1 —SOLUTION GUIDE—

Closed book: no textbook, no electronic devices, one sheet of paper with notes.

Read carefully before answering! Write your answers on the test paper and turn in your notes.

Question 1 *(5 points)*

What is the value of the last expression?

```
>>> L = ['new', 'york', 'times']
>>> M = ['washington', 'post'] + L
>>> L[2]
```

In other words, what is is the value of L[2] after L and M have been defined?

SOLUTION 'times' (or you can just say times, since python often omits quote marks when displaying output)

Rubric: (2 for 'times' or times) + (3 for just that, no list or other junk) **Assess:** [state]

Question 2 *(5 points)*

What is the value of the last expression?

```
>>> L = ['new', 'york', 'times', 'dot', 'com']
>>> M = range(1,6)
>>> L[M[2]]
```

SOLUTION 'dot' (or you can just say dot)

Rubric: (2 for dot + 3 for no list or other junk) **Assess:** [state]

Question 3 *(5 points)*

What is the value of the last expression?

```
>>> L = ['play', 'scrabble', 'now']
>>> M = L + ['or', 'go', 'swimming']
>>> L[:2] + M[5:]
```

SOLUTION ['play', 'scrabble', 'swimming']

Rubric: (1 for a list of words) + (1 for a list that includes at least play, scrabble, swimming) + (3 for correct list) **Assess:** nothing

Question 4 *(20 points)*

Write a trace of the call mystery(28, 36), for the function definition below. That is, starting with mystery(28, 36), show each call that is made, with the argument values, connected by arrows to show what call leads to what call. In addition, draw a reverse arrow labeled with the value returned, for each call.

```
def mystery(m, n):
    if m % n == 0:
        return n
    return mystery(n, m % n)
```

SOLUTION Here's a trace (but the arrows are missing, sorry).

```
mystery(28,36) returns 4
mystery(36,28) returns 4
mystery(28,8) returns 4
mystery(8,4) returns 4
```

Note: This is a famous program that computes the greatest common divisor.

Rubric: 5 for at least one call + 5 for most of the calls + 5 all of the calls + 5 correct result number
Assess: [execution]

Question 5 *(15 points)*
Consider this code. What does it print? (Hint: you might want to write a trace of function ind for yourself.)

```
def ind(item, lst):
    if lst == []:
        return None
    if item == lst[0]:
        return 0
    return 1 + ind(item, lst[1:])

L = range(7)
M = map(lambda x: x+2, L)
print M
print ind(5, M)
```

SOLUTION It prints

```
[2, 3, 4, 5, 6, 7, 8]
3
```

Rubric: (3 M is a list) + (2 M contains successive numbers) + (5 M has the right numbers) + (3 correct value from ind(5,M) for their M) + (2 value from ind(5,M) is 3)
Assess: execution

Question 6 *(25 points)*
Implement the following function.

```
def suffixes(L):
    '''Assume L is a list.
       Return the suffixes of L, in decreasing order by length.'''
```

For example, `suffixes([5, 1, 42, 3])` should return

```
[[5, 1, 42, 3], [1, 42, 3], [42, 3], [3], []]
```

Hint: As the example shows, we consider `[ ]` to be a suffix of any list, so `suffixes([ ])` should return `[[ ]]`.

SOLUTION

```
def suffixes(L):
    if L == []: return [[]]
    else:
        return [L] + suffixes(L[1:])
```

The brackets are important here. Don't just put brackets because you think something is a list. This is wrong:

```
...             return [L] + [suffixes(L[1:])]
```

It takes the list of suffixes and makes a one-element list out of that before catenating with [L]. This is also wrong:

```
...             return L + suffixes(L[1:])
```

Try it and see what happens.

Rubric: (5 syntax) + (5 correct base case) + (5 design using recursion, i.e., suffixes (L [1:]) combined with L) + (5 result has [L]) + (5 result has suffixes (L [1:]) not in brackets)
Assess: coding

FRIDAY PART OF THE TEST

Rubric: for Q7: (5 file loads without error) + + (6 works on the given test cases) + (6 works in general) + (3 uses recursion not filter) **Rubric:** for Q8: (3 uses filter) + (4 works on the given test cases) + (3 works in general)

If the solutions use additional functions, they are not required to have docstrings – but in the future we will require this. **Assess:** Q8 and Q9: coding

```
# test1_Q78sol - questions 7 and 8 on first exam, 2014 - SOLUTION GUIDE

##############################################################################
# RULES: you can use Moodle to download this file and upload your solution.
# You can use IDLE to edit and run your program.  You should NOT look at
# other programs in IDLE, you should NOT use any other programs, and you
# should NOT use any notes or books.
# According to the Honor Code, you should report any student who appears
# to be violating these rules.
##############################################################################

########################
# Question 7 (20 points)
# Implement this function, using recursion and not using filter.
########################

def remBad(strs):
    '''Assume strs is a list of strings.  Remove every occurrence of "bad"
    and leave the rest unchanged.  For examples, look at remBadTest() below.'''

    if strs == []:
        return []
    elif strs[0] == "bad":
        return remBad(strs[1:])
    else:
        return [strs[0]] + remBad(strs[1:])



# Examples and a function to help testing

test1 = ["bad"]
ans1 = []
test2 = ["cat"]
ans2 = ["cat"]
test3 = ["bad", "cat", "ate", "bad", "rat"]
ans3 = ["cat", "ate", "rat"]


def remBadTest():
    if remBad(test1)==ans1 and remBad(test2)==ans2 and remBad(test3)==ans3:
        print "success"
    else:
        print "test failed"


#########################
# Question 8 (10 points)
# Implement this function using the 'filter' function, and not recursion.
#########################

def remBadAlt(strs):
    '''same as remBad'''
    return filter(lambda s: s != "bad", strs)


def remBadAltTest():
    if remBadAlt(test1)==ans1 and remBadAlt(test2)==ans2 and remBadAlt(test3)==ans3:
        print "success"
    else:
        print "test failed"
```