CSE 101 - Introduction to Programming
End Sem Exam - Monsoon 2018
Max. Marks - 50 **Time - 2 hours**

Instructions:

1. You will be expected to write Python code in this exam. We recommend that you draw vertical lines to make your indentation clear.
2. Assume the use of Python3 in all of the questions below.
3. Write your details on both the question paper and the answer sheet. Only the answers written in the answer sheet will be evaluated. No extra sheets will be provided.
4. There are 3 sheets in this question paper printed both sides. There are a total of 6 questions and many questions have subparts. All subparts of a question must be attempted together. Mention the question number and the subpart number clearly in your answer sheet.

Question-1: This question has 5 subparts. Each subpart is for 2 marks. All questions below have binary grading, i.e., either 0 or full marks will be awarded. [5*2 = 10 Marks]

1.1. Examine the code below. What does *always_sunny(('cloudy'),('cold',))* evaluate to?

```
def always_sunny(t1, t2):
    """ t1, t2 are non empty """
    sun = ("sunny","sun")
    first = t1[0] + t2[0]
    return (sun[0], first)
```

1.2. What is the value of L after you run the code below?

```
L = ["life", "answer", 42, 0]
for thing in L:
    if thing == 0:
        L[thing] = "universe"
    elif thing == 42:
        L[1] = "everything"
```

1.3. What is the value of L3 after you execute all the operations in the code below?

```
L1 = ['re']
L2 = ['mi']
L3 = ['do']
L4 = L1 + L2
L3.extend(L4)
L3.sort()
del(L3[0])
L3.append(['fa','la'])
```

1.4. What is the value of brunch after you execute all the operations in the code below?

```
L1 = ["bacon", "eggs"]
L2 = ["toast", "jam"]
brunch = L1
L1.append("juice")
brunch.extend(L2)
```

1.5. What is printed when the below code is run?

```
mysum = 0
for i in range(5, 11, 2):
    mysum += i
    if mysum == 5:
        break
        mysum += 1
print(mysum)
```

Question-2: This question has 5 subparts. Each subpart is for 3 marks. All questions below have binary grading, i.e., either 0 or full marks will be awarded. [5*3 = 15 Marks]

2.1. a) How many total lines of output will show up if you run the code *below? [1 mark]*
b) What is the output? [2 marks]

```
def add(x, y):
    return x+y

def mult(x, y):
    print(x*y)

add(1,2)
print(add(2,3))
mult(3,4)
print(mult(4,5))
```

2.2. What does the following code print?

```
def f(s):
    if len(s) <= 1:
        return s
    return f(f(s[1:])) + s[0]  #Note double recursion
print(f('mat'))
print(f('math'))
```

2.3 Explain the functionality of the following code.

```
# Assume that n is greater than or equal to 1
def fun1(n):
    if(n == 1):
        return    0
    else:
        return    1 + fun1(n//2) + fun1(n//2)
```

2.4. Write a recurrence relation and time complexity for the following sorting algorithm.

```
def fun2(arr, s_index, e_index):
    if(s_index >= e_index):
        return
    # Assume that minIndex() returns index of minimum value in
    # array arr[start_index...end_index]
    m_index = minIndex(arr, s_index, e_index)
    arr[s_index],arr[m_index] = arr[m_index],arr[s_index]
    fun2(arr, s_index + 1, e_index)
```

2.5. Let s is a non-empty string, *len* be the length of the string s and *num* be the number of characters printed on the screen. Give the relation between *num* and *len* when the function abc is called.

```
def abc(s):
    if(len(s) < 1):
        return
    for i in range(4):
        abc(s[1:])
    print(s[0])
```

Question-3: This question has 2 subparts. [3+2=5 Marks]
3.1 Use recursion to implement the following function according to the specification.
3.2 Show your call stack trace for the example given in the specification.
Note: 3.2 will only be graded if the answer to 3.1 is correct. No marks will be awarded if the solution for 3.1 does not use recursion.

```
def filter(nlist):
    """Returns: A copy of nlist with all odd numbers removed,
    preserving order
    Example: filter([1, -1, 2, -3, -4, 0]) is [2, 4, 0]
    Precondition: nlist is a (possibly empty) list of numbers"""
```

Question-4: This question has 2 subparts. [2+3 = 5 Marks]
Here is an implementation of a function which is wrong as per the given specification.
4.1 Correct this implementation by modifying *exactly one statement*.
4.2 Show the trace of your corrected code for the inputs

- **thelist= [4, 7, -5, 70, 8, 70]**
- **limit = 1**

```
def first_below(thelist, limit):

    """
    Returns: index of leftmost item in thelist that has value less than
    limit. (Returns -1 if no such item exists in thelist.)

    Precondition: thelist is a possibly empty list of ints. limit is an
    int.

    Example input/output pairs:
    first_below([4, 2, 5, -2], 3) --> 1
    first_below([4, 10, 5,-2], 3) --> 3
    first_below([4, 2, 5,-2], 5) --> 0
    first_below([4, -2, 5, 2],-3) --> -1
    first_below([4, -2, 5, 2],-2) --> -1
    first_below([],5) --> -1
    first_below([4],5) --> 0
    """

    #code begins here

    j = 0
    while j+1 < len(thelist) and thelist[j+1] >= limit:
        j = j + 1
    if j+1 == len(thelist):
        return -1
    else:
        return j+1
```

Question-5: Given a list of numbers L as below:
L= [12, 34, 1, 45, -9, 7, 70, 35, 0, 8]
Show the final value of L and all the steps involved in executing the partition steps of Quicksort Algorithm (as discussed in the class). Assume that the last element is chosen as the pivot element. The list L is to be sorted in ascending order. (No need to show any algo/code.) [5 Marks]

Question-6: This question has 3 subparts. [2+3+5 = 10 Marks]
Assume that objects of class Course have two attributes:

- ***label [non-empty str]:*** unique identifying string, e.g., 'CSE101'
- ***prereqs [list of Course, maybe empty]:*** Courses that one must complete before this one.

6.1. Give a class definition header for the class Course. Course does have the Object class as its parent class. [2 Marks]

6.2. Give a constructor definition for the Course class which have a default value for *prereqs* parameter as empty list, i.e., [ ]. [3 Marks]

6.3 Read the description below carefully and write the corrected code for the function in your answer sheet. [5 Marks]

Consider the following header and specification of a non-method function.

```
def requires(c, other_label):
    """
    Returns: True if Course with label other_label must be taken before
    c, False otherwise.
    Precondition: c is a Course. other_label is a non-empty string.
    """
```

Example intended operation:
Suppose,

- ***c1 is a Course with label 'CSE101' and empty prereqs list***
- ***c2 is a Course with label 'CSE211' and prereqs list [c1]***
- ***c3 is a Course with label 'CSE280' and prereqs list [c1]***
- ***c4 is a Course with label 'CSE311' and prereqs list [c2, c3]***

Then, all of the following should evaluate to True.

- requires(c4, 'CSE280')
- requires(c4, 'CSE211')

And all of the following should evaluate to False:

- requires(c4, 'CSE101')
- requires(c1, 'CSE280')
- requires(c3, 'CSE280')

While a majority of the lines are correct, there is at least one error in the proposed implementation below. For each error, mention the erroneous line of code and write down/explain the correct version. No marks will be awarded if only the error is identified and no solution or incorrect solution is provided.

```
def requires(c, other_label):

    #Note to students - DO NOT alter the header.

    if len(c.prereqs) <= 1:
        return False
    else:
        for p in prereqs:
            if p.label == other_label:
                return True
            elif requires(other_label, c):
                return True
    return False
```
