Last Name: UTION First Name: SOL Cornell NetID: CS1110

# Solution: CS 1110 Prelim 2 April 26, 2016

1. [6 points] What is the output if the following module is run:

```
def F1(x):
    y = [] # empty list
    n = len(x)
    for k in range(n):
        if k==0 or k==n-1:
            y.append(x[k])
        else:
            y.append(x[k-1]+x[k+1])
    return y

def F2(x):
    y = x
    n = len(x)
    for k in range(n):
        if k==0 or k==n-1:
            y[k] = x[k]
        else:
            y[k] = x[k-1]+x[k+1]


if __name__ == '__main__':
    x = ["A","B","C","D"]
    z = F1(x)
    print x
    print z

    print "..."

    x = ["A","B","C","D"]
    z = F2(x)
    print x
    print z
```

Solution:

```
['A', 'B', 'C', 'D']            1 point
['A', 'AC', 'BD', 'D']          2 points (1 for  end entries, 1 for two middle entries)
...
['A', 'AC', 'ACD', 'D']         2 points (1 for each middle entry)
None                            1 point
```

2. [4 points] What is the output if the following module is run?

```
class C(object):
    def __init__(self,u,v):
        self.x = u
        self.y = v

    def Reflect1(self):
        temp = self.x
        self.x = self.y
        self.y = temp

    def Reflect2(self):
        temp = self.x
        self.x = self.y
        self.y = temp
        z = self
        return z

if __name__ == '__main__':
    x = 1
    y = 2
    P = C(1,2)
    P.Reflect1()
    print x,y
    print P.x,P.y

    x = 1
    y = 2
    P = C(1,2)
    Q = P.Reflect2()
    print P.x,P.y
    print Q.x,Q.y
```

*No partial credit without a diagram that displays objects and references.* You can put such a diagram in the whitespace to the right of the code above.

Solution:

```
1   2
2   1
2   1   # grading comment: should be the same as previous line
2   1   # grading comment: should also be the same as previous line
```

1 point per line

3. [11 points] Assume the availability of these classes:

```
class Point(object):
    """
    Attributes:
        x: float, the x-coordinate of a point
        y: float, the y-coordinate of a point
    """
    def __init__(self,x,y):
        """ Creates a point.
        PreC: x and y are floats
        """
        self.x = x
        self.y = y

class LineSeg(object):
    """
    Attribute:
         P1: endpoint [Point]
         P2: endpoint [Point]
    """
    def __init__(self,Q,R):
        self.P1 = Q
        self.P2 = R
```

(a) Write code that assigns to variable `Z` a reference to a `LineSeg` object that represents the line segment with endpoints (1,2) and (3,4).

Solution: 2 points total

```
A = Point(1,2)          # .5 point

B = Point(3,4)          # .5 point

Z = LineSeg(A,B)        # 1 point
```

(b) We say that a point $(a, b)$ is positive if $a > 0$ and $b > 0$. Write a boolean-valued *method* `Pos` for the `Point` class that returns `True` if the referenced point is positive. Your answer must include the header definition ("`def Pos(`...”). Omit the docstring.

Solution:

```
def Pos(self):
   return self.x > 0 and self.y > 0
```

```
3 points total:


+1: header has self parameter

+1 uses self.x and self.y

+1: everything else
```

(c) Complete the following function:

```
def F(L):
   """Returns a list of all those references in L that are to LineSeg objects
    whose endpoints are both positive.

    PreC: L is a list of references to LineSeg objects.
   """
```

You may assume the availability of the method `Pos` in part (b).

Solution:

```
K = []
for lineseg in L:
   if lineseg.P1.Pos() and lineseg.P2.pos()
       K.append(lineseg)
return K
```

`+1: references to the points in a LineSeg.`

`+1: correct calls to the method Pos for Points`

`+1: everything else .5 for K stuff and .5 for the for statement`

(d) Complete the following function. Nested loops are allowed.

```
def F(LofP1,LofP2):
   """Returns a list of references to LineSeg objects that encodes all possible
   line segments obtained by connecting a point in LofP1 to a point in LofP2

    PreC: LofP1 and LofP2 are nonempty lists of references to Point objects. Assume
    that the points encoded by these lists are all distinct.
   """
```

Solution:

`3 points`

```
    K = []
    for u in LofP1:
        for v in LofP2:
            K.append(LineSeg(u,v))
    return K
```

`+1: nested loop`

`+1: correct call to LineSeg`

`+1: everything else`

4. [6 points] Assume the availability of the following function:

```
    def randiList(L,R,n):
        """ Returns a length-n list of random integers from interval [L,R] inclusive.

        PreC: L,R,n ints with L<=R and n>=1
        """
```

Complete the following script so that it assigns to a float variable `p` an estimate of the probability that when you roll three dice, exactly two of them have the same value. Your solution must make effective use of the three generated lists of random integers created in the assignment to `D` below. No additional calls to `randiList` or `randi` are allowed. Don't print or return `p`.

If you choose to use a helper function, put its definition here, *before* your Application script code:

```
if __name__ == '__main__':
    """ Roll 3 dice many times and record the outcome using lists.
    """
    N = 1000000
    D = [randiList(1,6,N), randiList(1,6,N), randiList(1,6,N)]
    # Each of the three lists in D is the record of the N rolls of one of
    # the three dice.

    # Hint: assign to a variable m the number of times that EXACTLY two
    # of the dice have the same value.
```

Solution:

```
    m = 0
    D0 = D[0]
    D1 = D[1]                       # 2 points for referencing the 3 lists via D
    D2 = D[2]
    for k in range(N):
        B0 = D0[k]==D1[k] and D0[k] != D2[k]
        B1 = D1[k]==D2[k] and D1[k] != D0[k]   # 2 points for boolean stuff
        B2 = D2[k]==D0[k] and D2[k] != D1[k]
        if B0 or B1 or B2:
            m+=1
    p = m/float(N)                             # 2 points for p  (-1 if m/N)
```

```
###### Alternate solution: uses helper above #####


def exactly_two(v1, v2, v3):
    """Returns 1 if exactly two of the (int) args are the same, 0 o.w."""
    if (v1==v2 and v1!=v3) or (v1!=v2 and v2==v3) or (v1==v3 and v1 != v2):     # 2 points
        return 1
    else:
        return 0



    m = 0
    for ind in range(N):
        m+= exactly_two(D[0][ind], D[1][ind], D[2][ind])     # 2 points
    p = m/float(N)                                           # 2 points
```

5. [6 points] These classes were involved in Assignment 6:

```
class Speech(object):
    """
    attributes:
        theSpeaker       the name of the speaker [str]
        lines            each item is a (file) line in the speech [list of str]
    """
class Scene(object):
    """
    attributes:
        actNumber       the number of the act [int]
        sceneNumber     the number of the scene [int]
        location     the location of the scene [str]
    """

class Play(object):
    """
    attributes:
        theTitle        the name of the play [str]
        theSpeeches     a list of all the speeches in the play [list(Speech)]
        theScenes       a list of all the scenes in the play [list(Scene)]
        nLines          the total number of lines in the play [int]
    """
```

Complete the following function so that it performs as specified. Hint: you don't need `stringToWordList`.

```
def qMarks(playlist):
  """ Returns the total number of question marks that occur amongst all the
  lines in all the speeches in all the plays referenced by playlist.

  PreC: playlist is a list of references to Play objects.
  """
```

Solution:

```
n = 0
for p in playlist:
    # p is a reference to a Play object
    for s in p.theSpeeches:
        # s is a reference to a speech object
        for line in s.lines:
            # line is a string
            n += line.count("?")
return n
```

```
1 pt first loop iterates over plays
1 pt second loop iterates over speeches
1 pt third loop iterates over lines
```

```
2 points if proper access to attrinbutes
1 point for n and count

-1 if return missing or improper
```

6. [1 point] Write your last name, first name, and Cornell NetID at the top of *each* page, and circle your lab time on the first page.