Computer Science 1 — CSci 1100
Test 2 Overview and Practice Questions
Fall Semester, 2016

Important Logistical Instructions:

- Test 1 will be held **Monday, October 24, 2016**.
- Most students will take the exam from 6:00 - 7:30 pm. Students who provided Professor Stewart or Professor Turner with an accommodation letter indicating the need for extra time or a quiet location will take the exam starting at 4:30 pm.
- Room assignments will be posted on Submitty by Thursday night, October 20. For most students these assignments **will be different**.
- Students MUST:
  - **Go to their assigned rooms.**
  - **Bring their IDs to the exam.**

  Failing to do one of these will result in a **20 point** penalty on the exam score. Failure to do both will cost **40 points**.

Solutions

The following are the solutions to the practice problems. Please be aware that there may be more than one way to solve a problem and so your answer may be correct despite being different from ours.

Questions

1. Write a Python function called `compare_date` that takes as arguments two lists of two integers each. Each list contains a month and a year, in that order. The function should return -1 if the first month and year are earlier than the second month and year, 0 if they are the same, and 1 if the first month and year are later than the second. Your code should work for any legal input for month and year. Example calls and expected output are shown below:

   ```
   >>> compare_date( [10,1995], [8,1995] )
   1
   >>> compare_date( [5,2010], [5,2010] )
   0
   >>> compare_date( [10,1993], [8,1998] )
   -1
   ```

   Solutions:

   ```
   def compare_date(x, y):
       if x[1] > y[1]:
           return 1
       elif x[1] < y[1]:
           return -1
       elif x[0] > y[0]:
           return 1
       elif x[0] < y[0]:
           return -1
       else:
           return 0
   ```

   Second version, combining the logic:

   ```
   def compare_date(x, y):
       if x[1] > y[1] or (x[1] == y[1] and x[0] > y[0]):
           return 1
       elif x[1] < y[1] or (x[1] == y[1] and x[0] < y[0]):
           return -1
       else:
           return 0
   ```

2. Assume `v` is a list containing numbers. Write Python code to find and print the highest two values in `v`. If the list contains only one number, print only that number. If the list is empty, print nothing. For example, if we assigned

   ```
   v = [ 7, 3, 1, 5, 10, 6 ]
   ```

   then the output of your code should be something like

   ```
   7 10
   ```

   If we are given that

   ```
   v = [ 7 ]
   ```

   then the output of your code should be

   ```
   7
   ```

   Solution:

```
v.sort()
if len(v) == 1:
    print(v[0])
elif len(v) > 1:
    print(v[-2], v[-1])
```

3. Consider a simplified version of the lab Yelp data, where just the name of the restaurant, the type of restaurant, and the ratings are provided. Assume these values **have already been read into a list of lists** of the form below:

```
  restaurants = [ [ 'Acme', 'Italian', 2, 4, 3, 5],
                  [ 'Flintstone', 'Steak', 5, 2, 4, 3, 3, 4],
                  [ 'Bella Troy', 'Italian', 1, 4, 5] ]
```

   Write a segment of Python code that prints all `Italian` restaurants in the `restaurants` list that have no ratings of value 1 and at least one rating of value 5. In the above example, `Acme` would be printed in the output, but `Flintstone` and `Bella Troy` would not. `Flintstone` is not Italian and `Bella Troy` has a 1 rating. Your code should work for any legal version of `restaurants`.

   Solution:

```
for rest in restaurants:
    if rest[1] == 'Italian' and (1 not in rest[2:]) and (5 in rest[2:]):
        print(rest[0])
```

4. Continuing with the Yelp data, assume that you have the code

```
in_file = open('yelp.txt')

for line in in_file:
    p_line = parse_line(line)
    print(p_line)
```

   and that the `parse_line` function will return a list that looks like

```
["Meka's Lounge", 42.74, -73.69, "Bars", [5, 2, 4, 4, 3, 4, 5], 3.857142857142857 ]
```

   where the last entry in the list is the average rating. Modify the `for loop` above to create a list called `high` that stores the names of all restaurants that have an average rating of at least 4.0. You do not have to print `high`.

   Solution: Don't forget to initialize `high` to be the empty list:

```
in_file = open('yelp.txt','r')

high = []
for line in in_file:
    p_line = parse_line(line)
    if p_line[5] >= 4.0:
        high.append( p_line[0] )
```

5. In the game of chess you can often estimate how well you are doing by adding the values of the pieces you have captured. The pieces are Pawns, Bishops, Knights, Rooks and Queens. Their values are

```
    P - (P)awn, value = 1
    B - (B)ishop, value = 3
    K - (K)night, value = 3
    R - (R)ook, value = 5
    Q - (Q)ueen, value = 9
```

Write a Python function called `chess_score` that takes a single string as an argument and returns the combined values represented by the pieces in the string. You may assume that only `'P'`, `'B'`, `'K'`, `'R'`, and `'Q'` appear in the string. You may **not** use any if statements and you may **not** use any loops. As an example,

```
print(chess_score('BQBP'))
```

should output the value 16 because there are 2 Bishops (3 points each), 1 Queen (9 points each), and 1 Pawn (1 point each).

Solution:

```
def chess_score(pieces):
    pawns = pieces.count('P')
    bishops = pieces.count('B')
    knights = pieces.count('K')
    rooks = pieces.count('R')
    queens = pieces.count('Q')
    return pawns + 3*bishops + 3*knights + 5*rooks + 9*queens;
```

6. Write a function called `in_between` that takes a list as a parameter and returns a new list that contains only the numbers that are greater than the first value in the list (at index 0) and less than the last value in the list. As an example

```
print(in_between( [2, 3, 6, 5, 1, 6] ))
```

should output

```
[3, 5]
```

Solution:

```
def in_between(v):
    t = []
    for x in v:
        if v[0] < x and x < v[-1]:
           t.append(x)
    return t
```

7. You are given a file that contains, on each line of input, three integers separated by commas. Write a Python program that sums all of the first integers, the second integers, and the third integers, outputting the resulting sums all on one line, separated by commas. As a simple example, if the input is

```
 2, 5,7
 3,  6,   10
   1, 2, -3
  2, 4, 1
```

Then the output should be

```
8, 17, 15
```

Solution: The following solution does not require the creation of three lists to store the values prior to summation.

```
sums = [0,0,0]

for line in open('infile.txt'):
    dat = line.split(',')
    sums[0] += int(dat[0])
    sums[1] += int(dat[1])
    sums[2] += int(dat[2])

print("{}, {}, {}".format(sums[0], sums[1], sums[2]))
```

The second solution does create three lists. The first solution is slightly preferred because of this, although you would not lose points on a test unless we explicitly disallowed the creation of the three lists.

```
firsts = []
seconds = []
thirds = []

for line in open('infile.txt'):
    dat = line.split(',')
    firsts.append(int(dat[0]))
    seconds.append(int(dat[1]))
    thirds.append(int(dat[2]))

print("{}, {}, {}".format(sum(firsts), sum(seconds), sum(thirds)))
```

8. Write Python code to generate the following ranges

   (a) $[100, 99, 98, \dots, 0]$

   (b) $[55, 53, 51, \dots, -1]$

   (c) $[3, 5, 7, 9, \dots, 29]$

   (d) $[-95, -90, -85, \dots, 85, 90]$

   Solutions: Please note that this question came from an old exam using Python 2.7 where the `range` function returned a list. In Python 3.5, which we are using, we need to convert to a list.

```
# Part a
list(range(100,-1,-1))

# Part b
list(range(55,-2, -2))  # could have 55, -3, -2 as well

# Part c
list(range(3,30,2))  # 30 could be replaced by 31

# Part d
list(range(-95,91,5))  # 91 could be replaced by any of 91..95
```

9. Write a **while** loop to add all of the numbers in a list `v` until it reaches a negative number or until it reaches the end of the list. Store the sum in the variable `result`. Your code should work for any version of `v` containing only numbers. For example, the value of `result` should be 25 after the loop for both of the following lists:

```
  v = [ 10, 12, 3, -5, 5, 6 ]

  v = [ 0, 10, 3, 6, 5, 1 ]
```

   Solutions:

```
i = 0
result = 0
while i < len(v):
    if v[i] < 0:
        break
    result += v[i]
    i += 1
```

Version combining logic

```
i = 0
result = 0
while i < len(v) and v[i] >= 0:
    result += v[i]
    i += 1
```

10. Write Python code that takes a list of numbers, `v`, and outputs the *positive* values that are in `v` in increasing order, one value per line. If there are no positive values, then the output should be the string `'None'`. You may assume there is at least one value in the list. As an example,

```
v = [ 17, -5, 15, -3, 12, -5, 0, 12, 22, -1 ]
```

Then the output of your code should be

```
12
12
15
17
22
```

As a second example, if

```
v = [ -17, -5, -15, -3, -12, -5, 0, -12, -22, -1 ]
```

then then output should be just

```
None
```

Solution:

```
v.sort()

if v[-1] <= 0:   # largest number is not positive
    print('None')
else:
    for x in v:
        if x > 0:
            print(x  )
```

Here is a second solution that involves a second list and two loops:

```
pos = []
for x in v:
    if x > 0:
        pos.append(x)
pos.sort()
```

```
if len(pos) == 0:
    print('None')
else:
    for x in pos:
        print(x)
```

11. What is the output of the following operations:

```
>>> mylist = [1,4,8,12,6]
>>> x = mylist.sort()
>>> print(x)

>>> mylist = [1,4,8,12,6]
>>> slice1 = mylist[2:4]
>>> slice1[0] = 20
>>> print(slice1)

>>> print(mylist)
```

Solution: Please test them for yourself.

12. Write a Python `for loop` to print out the values from the list `v` that are positive (0 is NOT a positive number).

Solution: This assumes that list `v` already exists

```
for x in v:
    if x > 0:
        print(x)
```

13. What is the output of the following program?

```
def spam(a1,b1,a2,b2):
    if (a1 == a2) and (b1 > b2):
        return 1
    else:
        return 0

def egg(a1,b1,a2,b2):
    if (a1 > a2) and (b1 == b2):
        return 0
    else:
        return 1


a1 = 3
b1 = 4
a2 = 6
b2 = 4

print(spam(a2, b2, a1, b1))
print(egg(a1, b1, a2, b2))

c = spam(a1, b2, a2, b1)

print(c)

c += egg(a1, b2, a2, b1)

print(c)
```

Solution: Please test them for yourself.

14. Write a function called `copy_half` that takes the name of two files as arguments. The function should copy the first, third, fifth, etc. lines (i.e. odd lines only) from the first file to the second file. For example, if the file names are `'in.txt'` and `'out.txt'` and if `'in.txt'` contains

```
starting line
  not this line
middle line is here
    skip this line too
     I like this line
```

then after the call

```
copy_half( 'in.txt', 'out.txt' )
```

the file `'out.txt'` should contain

```
starting line
middle line is here
     I like this line
```

Solution:

```
def copy_half( in_name, out_name ):
    inf = open(in_name)
    outf = open(out_name,'w')
    i = 0
    for line in inf:
        i += 1
        if i%2 == 1:
            outf.write(line)
    outf.close()
```

Solution using a list

```
def copy_half( in_name, out_name ):
    inf = open(in_name)
    lines = []
    for line in inf:
        lines.append(line)

    outf = open(out_name,'w')
    for i in range(0,len(lines),2):
        outf.write(lines[i])
    outf.close()
```

15. Write a segment of code that reads integers from a file called `test2.txt` and stores the positive values in one list, the negative values in a second list, and skips blank lines and zeros. The order of the values in each list should match the order of the input. Each line of input will contain either spaces or spaces and an integer. For example, if `test2.txt` contains

```
      11
  -3

  5
    0
```

Then after your code, the list `P` should be `[ 11, 5 ]` and the list `N` should be `[ -3 ]`.

Solution:

```
P = []
N = []
for s in open("test2.txt"):
    s = s.strip()
    if len(s) > 0:
        i = int(s)
        if i > 0:
            P.append(i)
        elif i < 0:
            N.append(i)
```

16. Give the output of each of the following

(a)

```
i = 4
L = [ 0, 12, 3, 5, 2, -1 ]
while 0 <= i and i < len(L):
    if L[i] < 0:
        break
    else:
        i = L[i]
    print(i, L[i])
```

Solution:

```
2 3
3 5
5 -1
```

(b)

```
tough = 2
for i in range(2):
    s = 1
    for j in range(i, tough):
        s += tough
    print(s)
    print(tough)
    tough = s
print(tough)
```

Solution:

```
5
2
21
5
21
```

17. Please show the output from the following code?

```
def get_min(v):
    v.sort()
    return v[0]

def get_max(v):
    x = max(v)
    return x

v = [ 14, 19, 4, 5, 12, 8 ]
if  len(v) > 10 and get_min(v) > 6:
    print("Hello")
    print(v[0])
    print(v[4])
else:
    print("So long")
    print(v[0])
    print(v[-1])
    if len(v) < 10 or get_max(v):
        print(get_max(v))
        print(v[0])
        print(get_min(v))
        print(v[0])
```

Solution: The trick to getting this completely right is to realize that the first `if` conditional never calls the function `get_min`. The first half of the test, `len(v) > 10` is `False` so the `and` can never be `True`, which means that Python does not bother to evaluate the second half of the logical expression — `get_min(v) > 6`.

```
So long
14
8
19
14
4
4
```

18. Show the output from the following code:

```
def elephant(height):
    time_step = 1
    steps = 0
    while steps < height:
        steps += time_step
        steps -= time_step//3
        time_step += 1
    print("{}, {}".format(time_step, steps))

elephant(0)
elephant(5)
elephant(6)
```

Solution:

```
1, 0
4, 5
5, 8
```

19. Show the output of the following code. Make sure we can determine what is output and what is scratch work.

```
def remove_something(z):
    z.remove( z[z[0]] )

v = [ 1, 8, [12, 8], 'hello', 'car' ]
x = 'salad'

if len(v[2]) >= 2:
    if x > v[3]:
        print( 'One')
        if v[0] == 1:
            print('Three')
    else:
        print('Two')
elif len(v) == 5:
    print('Six')
else:
    v.append('five')
    print('Ten')

remove_something(v)
print(v[1])
print(v[2])
v.append(x)
print(len(v))
```

Solution:

```
One
Three
[12, 8]
hello
5
```

20. You are given a list of lists represented as an NxN grid in which each list corresponds to one row of the grid. For example, a 4x4 grid is given by:

`x = [[1,2,3,4],[4,3,2,1],[2,1,4,2],[2,1,4,5]]`

Write a piece of code to print the grid in the following format with a vertical and horizontal line right in the middle:

```
1 2 | 3 4
4 3 | 2 1
----|----
2 1 | 4 2
2 1 | 4 5
```

Solution:

```
n = len(x)
for i in range(n):
    l = ''
    for j in range(n):
        l += str(x[i][j]) + ' '
        if j == (n//2-1):
            l += '| '
    print(l)
    if i == (n//2-1):
         print('-'*(n - n%2) + '|' + '-'*n)
```

21. Write a piece of code that repeatedly asks the user for numbers using `input` until the user enters 'stop'. Then, the program reports the sum of the values entered by the user and the total number of values strictly greater than zero. You can assume that the user enters a valid number until she enters stop.

    An example run of this code is given below.

```
Enter a value ==> 1.2
Enter a value ==> 0
Enter a value ==> 2
Enter a value ==> -1
Enter a value ==> stop
Sum: 2.2
Values > 0: 2
```

Solution:

```
sum = 0
numgt = 0
while (True):
    val = input('Enter a value ==> ')
    if val == 'stop':
        break
    val = float(val)
    if val > 0:
        numgt += 1
    sum += val
print("Sum:", sum)
print("Values > 0:", numgt)
```

22. Write a function `remove_val(l,val)` that removes all copies of `val` from list `l`.

    Suppose you are given a variable `x` containing numbers as shown below:

    `x = [1, 4, 2, 1, 2, 4, 4, 2, 5, 5, 2]`

    Then, your function should work as follows:

```
>>> remove_val(x,4)
>>> x
[1, 2, 1, 2, 2, 5, 5, 2]
```

Note: if your function returns a new list with this content instead of modifying it as given, you will lose points.

Solution:

```
def remove_val(l,val):
    while l.count(val) > 0:
        l.remove(val)
```

23. Suppose you are given the scores of two athletes in various competitions, provided as two separate lists. Assume there are unknown number of competitions numbered 1,2,3, etc. and the length of the two lists is the same.

    ```
    a1 = [11,8,11,9]
    a2 = [11,9,8,12]
    ```

    For example according this to list, both athletes got a score of 11 in competition 1. Print the index of all the competitions in which `a2` did better. For example, for the above lists, we would print:

    ```
    a2 is better in 2 4
    ```

    If there is no value in which `a2` is better, then you should print:

    ```
    a2 is never better
    ```

    Solution:

    ```
    found = False
    line = "a2 is better in "
    for i in range(len(a1)):
        if a2[i] > a1[i]:
            line += str(i+1) + ' '
            found = True
    if not found:
        line = "a2 is never better"
    print(line)
    ```

24. What is the output from the following code:

    ```
    >>> L1 = ['cat', 'dog', 'hawk', 'tiger', 'parrot']
    >>> print(L1[1:-1])
    >>> print(L1[1:-2])
    >>> print(L1[1:-4])
    >>> print(L1[1:0])
    >>> print(L1[1:10])
    >>> print(L1[::-1])
    >>> print(L1[1:4:2])
    >>> print(L1[::-2])
    ```

    Solution: Enter it into the python interpreter yourself!

25. What is the output of the following programs:

Part a

```
a = 25
b = 11
while True:
    print(a, b)
    if a <= 0 or b <= 0:
        break
    if a > b:
        a = a - b
    else:
        b = b - a
    b -= 1
    a += 1
```

Solution:

```
25 11
15 10
6 9
7 2
6 1
6 0
```

Scratch area:

Part b

```
mylist = [10, -5, 4, 8, 1000, -1, -120, 18, 5.2]
for item in mylist:
    if item < 0:
        continue
    print(item)
```

Solution:

```
10
4
8
1000
18
5.2
```

Scratch area:

Part c

```
def spam(l,s):
    m = len(s)//2
    s1 = s[:m]
    s2 = s[m:]
    if l.count(s1) == 0:
        l.append(s1)
    if l.count(s2) == 0:
        l.append(s2)


l = ['ab','cd','de','fg']
s1 = 'abcde'
s2 = 'fghi'
spam(l,s1)
print(s1)


l = spam(l,s2)
print(s2)
print(l)
```

Solution:

```
abcde
fghi
None
```

Scratch area: