22C:16 CS:1210 Exam 2

April 4th, 6:30 pm to 8:30 pm

Instructions:

- This is an open notes exam and you have 2 hours to complete it. There are 4 problems in the exam and these appear on 6 pages. The exam is worth 130 points (13% of your grade).
- Make sure that you do not have any electronic devices (laptops, phones, calculators, dictionaries, etc.) on your desk; you are not allowed to access these during your exam.
- Write as neatly as you can.
- Show all your work, especially if you want to receive partial credit.

Name:

Circle your section:

SCA (M, W evening) A01 (8:30-9:20) A02 (11-11:50 MH) A03 (11-11:50 SH)

A04 (12:30-1:20) A05 (2:00-2:50) A06 (5:00-5:50)

---

1. **(40 points)** Evaluate these expressions. For parts (h) and (i) suppose that

   ```
   L = [100, ["hello", "bye"], 1000, [[1, 2], [2, 3], [3, 4]], 1000, 900L].
   ```

   For part (j) suppose that

```
J = [["This", ["is", "a"]], "nested", ["list", "with"], ["several", [["different"]]], ["levels", "of"], "nesting"].
```

   (a) `range(2, 11, 3)`

   (b) `[y-x for x in range(3) for y in range(x, 4)]`

   (c) `"".join("ok sure".split())`

(d) `range(5)[::-2]`

(e) `(range(5)+range(5)[::-1]).count(1)`

(f) `"".join([chr(ord(x)+1) for x in "best"])`

(g) `" San Francisco".strip().startswith("San")`

(h) `L[3].index([2,3])`

(i) `[y for x in L if type(x) == list for y in x]`

(j) `[x.split("ste") for x in J if "nest" in x]`

2. **(30 points)** In this problem you will execute code by hand and figure out what output is produced.

   (a) Write down the output produced when the following function is called as
   `parseSentences(["This is ok", "But this is not"])`

```
def parseSentences(L):

    wordList = ["".join(x.split()) for x in L]
    print wordList

    rotatedList = ["".join([chr(ord(y)+1) for y in x])
                      for x in wordList]
    print rotatedList
```

   (b) Write down the output produced when the following function is called as
   `removeOutliers([15, 2, 5, 1, 4, 15])`

```
def removeOutliers(L):
    mean = sum(L)/float(len(L))

    for i in range(len(L)):
        if L[i] > 2*mean:
            del L[i]

        print L

        if i > len(L)-1:
            break
```

3. **(30 points)** In this problem you will complete partially specified code.

   (a) In the partially completed code below, I provide a function called `processFile`. This function is required to read a given file, line by line and ignore any line containing a number (strictly) greater than 100. In addition, the function is required to store the numbers in each of the remaining lines in lists and return one large list containing all of these lists.

   For example, if the input file is

```
300, -10, -10
12 , 16, 99
-22, 100     , 100
1000, 1, 2, 3, 5
```

   then the function should return

```
[[12, 16, 99], [-22, 100, 100]]
```

   Fill in the three blanks in the code below to complete it.

```
def processFile(filename):
    # Open the given file
    f = open(filename)

    # Initialize masterList
    masterList = []

    # Process each line in the file
    for line in f:
        # Extract all the integers in the line into a list L

        L = ________________________________________________

        # Use a list comprehension to construct a boolean list
        # corresponding to L indicating which elements are > 100.
        # For example if L = [10, 200, 11] then the constructed
        # boolean list will be [False, True, False].

        discardList = ________________________________________

        # If there are no True values in discardList then
        # append L to the masterList

        if ____________________________________:

            masterList.append(L)

    return masterList
```

(b) Two strings s_1 and s_2 are said to be *similar* if (i) s_2 is obtained from s_1 by replacing one character by another or (ii) s_2 is obtained from s_1 by the deletion or the insertion of a single character. For example, strings `"sweet"` and `"tweet"` are similar by rule (i) and strings `"sweep"` and `"seep"` are similar by rule (ii).

Below, I have partially implemented a function that determines if a given pair of strings are similar. Complete the three blanks in this function.

```
# Returns True if s2 is obtained from s1 by substituting exactly one
# character for another or by deleting exactly one character or by
# inserting exactly one character.

def similar(s1, s2):
    # If the difference in lengths of s1 and s2 is
    # more than 1, return False
    if abs(len(s1) - len(s2)) > 1:
        return False

    # Check if s2 is obtained from s1 with one substitution
    if len(s1) == len(s2):
        for i in range(len(s1)):
        # Check if s1 without the i-th character equals
        # s2 without the i-th character

            if ________________________________________:

                return True

    # Check if s2 is obtained from s1 by one deletion
    if len(s1) > len(s2):
        for i in range(len(s1)):
            # Check if s1 without the i-th character equals s2

            if ________________________________________:

                return True

    # Check if s2 is obtained from s1 by one insertion
    if len(s2) > len(s1):
        for i in range(len(s2)):
            # Check if s2 with the i-th character removed equals s1

            if ________________________________________:

                return True

    return False
```

4. **(30 points)** Write a function called `maxSumSubsequence` that takes as parameter a list `L` of integers and returns a contiguous slice of `L` whose sum is maximum among all contiguous slices. For example, the function call

```
maxSumSubsequence([-1, 2, 3, -2, 4, -10])
```

should return the slice `[2, 3, -2, 4]` because this contiguous slice sums to 7 and no other (contiguous) slice has a larger sum.
Note: My code for this is 8 lines long, yours should not be too much longer.