5LN715 Written Exam 2020-03-27

Instruction:

Submit the solutions as one pdf-document in the Student Portal (as an assignment). Deadline 2020-03-27 20.00. **By submitting the solution you certify that it is the product of your own creativity and skills. You are not allowed to take help or work with anyone else.** However, you can use available online and printed material.

Only short answers are necessary, but should be exact, and neatly and clearly formatted, as details matter. Use of Python 3 is presupposed. Remember that indentation is significant! Of course, different solutions are often possible. Try to find the best ones!

The tasks are intended to be clear, well-defined, and straightforward. However, if some task seems ambiguous or open to you, solve it in the way you find most reasonable (most relevant given the course content), and write a note about how you interpreted the instruction. We will also monitor the Student Portal forum during the morning. You are welcome to ask for clarifications, but please don't reveal the solutions you have in mind.

The grade thresholds are as follows: G: 50% and VG: 75%. (The examination of the course also involves a series of assignments, as you know.)

GOOD LUCK!

Question 1 [max 6 p]

Here is some code handling blogs, blog posts and comments on blogs. A blog has an owner who writes all the posts. The posts have a title and some text. There can be comments on blog posts, and also comments on other comments. A comment has an author and a text.

```
class Blog:
    def __init__(self, name, owner):
        self.name = name
        self.owner = owner
        self.posts = []

    def add_post(self, post):
        self.posts.append(post)

    def output_with_comments(self):
        title = f'{self.name} (by {self.owner})'
        print(title)
        print('=' * len(title))
        print()
        for p in self.posts:
            p.output_with_comments()
        print()
```

```
class BlogPost:
    def __init__(self, title, text=''):
        self.title = title
        self.text = text
        self.comments = []

    def add_comment(self, comment):
        self.comments.append(comment)

    def output_with_comments(self):
        print(self.title)
        print('-' * len(self.title))
        print(self.text)
        print()
        for c in self.comments:
            c.output_with_comments(level=1)
        print()


class Comment:
    indent_marker = '> '

    def __init__(self, author, text=''):
        self.author = author
        self.text = text
        self.comments = []

    def add_comment(self, comment):
        self.comments.append(comment)

    def output_with_comments(self, level=0):
        indent = self.indent_marker * level
        print(f'{indent}[Comment by {self.author}:]')
        print(f'{indent}{self.text}')
        for c in self.comments:
            c.output_with_comments(level=level+1)
```

The `output_with_comments` method on a blog might print something like the text in the following box. (If this was real it would probably output HTML suitable for the web instead, but for now it is just plain text.)

```
My Thoughts (by Huey)
=====================

New blog
--------
I've started a blog. This is the first post.

> [Comment by Dewey:]
> I look forward to reading your blog!

Thought for today
-----------------
Flowers are nice. Do you also like them?

> [Comment by Dewey:]
> I agree.
> [Comment by Louie:]
> That is so wrong!
> > [Comment by Dewey:]
> > You are so negative, Louie!
> > > [Comment by Louie:]
> > > I just don't agree. Flowers are bad!
> [Comment by Huey:]
> I forgot to say that I especially like sunflowers.

Ending the blog
---------------
This is my last entry here. It has been fun!

> [Comment by Louie:]
> Actually I liked your blog.
```

(a) Name one *class variable* in this example? **(1 p)**

(b) In the example output, for which shown objects is it true that the length of its instance variable `comments` is 1? **(1 p)**

(c) Write a `posts_with_text` method for Blog which returns a list of blog posts in that blog where the title or the text of the post contains that (exact) text. **(2 p)**

(d) There should be a variant of `BlogPost` called `BlogPostForbiddingComments` which is like a normal blog post, except that if you try to add a comment to it it should yield an error by doing

```
raise TypeError
```

Write the full code for that, inheriting from the given class! **(2 p)**

Question 2 [max 4×2 p]

Determine the worst case time complexity for each code snippet below, and support your answer with a short argument. Assume that `numbers` is an array (of length n) containing integers and that `m` is a positive integer.

(a)

```
for i in numbers:
    print(i)
    for j in range(m):
        print(m)
    for k in range(50000):
        print(k*i)
for i in numbers:
    print("hello world")
```

(b)

```
i = len(numbers)
while i > 0:
    numbers[i] = numbers[i]*5
    i -= 3
```

(c)

```
for j in numbers:
    i = len(numbers)
    while i > 0:
        print(j)
        i = round(i/2.8)
```

(d)

```
for i in range(len(numbers)):
    for j in range(i):
        k = numbers[i] + numbers[j]
```

Question 3 [max 10 p]

Write code for the two methods

```
printSkip(self)
removeValue(self, value)
```

that should be part of the LinkedList class below, which represent a singly-linked list. Each node has an integer representing a *skip*, and a value which should be printed, called data. You are not allowed to change the given code.

The `printSkip` method should print the value in the first node, then skip as many nodes as indicated by the `skip` in the first node, print the next value, and skip the number of nodes indcated by that `skip`, and so on. When it reaches the end of the list, it should just stop.

`removeValue` takes a value as input, and should remove all nodes containing that value stored in `data`.

Note that each method should go through the list only one time, i.e. have time complexity $O(n)$. For full points, we expect as simple a solution as possible (without any unnecessary if-clauses, for instance).
As an example, a linked list containing the nodes (data, skip):

```
("a",2) -> ("b",5) -> ("c",1) -> ("d", 3) -> ("e",6) -> ("a",1) -> ("g",6) -> ("a", 3)
```

Should printSkip "a, c, d, g" and if we apply `removeValue` with "a", it should result in this list:

```
("b",5) -> ("c",1) -> ("d", 3) -> ("e",6) -> ("g",6)
```

```
#Node class:
class Node:
    def __init__(self, data, skip, next):
        self.next = next
        self.data = data
        self.skip = skip

#List class
class LinkedList:
    def __init__(self):
        self.head = None

    def insert(self, data, skip):
        self.head = Node(data, skip, self.head)
```

Question 4 [max 10 p]

Below you will find the timings of two different sorting algorithms, sortX and sortY, for arrays of lengths 1,000, 10,000, and 100,000. The arrays contain integers from 0 to length-1, and have been manipulated before sorting using either of two strategies `shuffle1` or `shuffle2` outlined below. You should discuss the complexity and other features of these two sorting algorithms, given the times in these different settings. Make a guess and motivate it, about which sorting algorithms you think were used. (Note, the discussion and motivation is much more important than guessing correctly).

Timings:

```
sortX, shuffle1, 1000, time: 0.032445
sortY, shuffle1, 1000, time: 0.003253
sortX, shuffle2, 1000, time: 0.001209
sortY, shuffle2, 1000, time: 0.002829
sortX, shuffle1, 10000, time: 3.270695
sortY, shuffle1, 10000, time: 0.044512
sortX, shuffle2, 10000, time: 0.015209
sortY, shuffle2, 10000, time: 0.037252
sortX, shuffle1, 100000, time: 331.031591
sortY, shuffle1, 100000, time: 0.558342
sortX, shuffle2, 100000, time: 0.106189
sortY, shuffle2, 100000, time: 0.443340
```

Shuffling strategies:

```
#list initialization:
myList = list(range(length))

def shuffle1(l):
    for i in range(len(l)):
        r = randint(0,len(l)-1)
        (l[i],l[r]) = (l[r],l[i])


def shuffle2(l):
    for i in range(10):
        r = randint(0,len(l)-1)
        s = randint(0,len(l)-1)
        (l[s],l[r]) = (l[r],l[s])
```

Question 5 [max 10 p]

Assume that you want to write a program that will process a very large file of word-POS pairs. The output of the program should be a frequency list of each word-POS pair, with the percentage of a POS among all options for that word. For example, if the input file contains the word *book* 10 times, 8 times as a noun and 2 times as a verb, and the word *pen* 3 times, always as noun, your program should output: "book NOUN 8 (80%), book VERB 2 (20%), pen NOUN, 2 (100%)". The output should be sorted alphabetically based on the word and POS. The input format of the file to read is not really important for the task, but it makes it easy to extract word+POS pairs. The file will be very large, containing a high number of both types and tokens. You can assume that the distribution of words and POS-tags are representative of English.

Discuss at least two different options for data structures that you could use for solving this task. Try to come up with as efficient data structure options as you can. Describe how you would store the information using each data structure. Then, go through all operations needed

for storing the data and printing the asked for information. For each operation, discuss what the time complexity is of that operation with each of your data structure options. Note that the exact operations might be slightly different for the two data structures. Also note that it is possible, but not necessary, that your proposed data structure solutions use more than one data structure each.

Note: you do not have to write any code here. Only describe how you would have organised the data structures used in your code, and the time complexity issues.

Question 6 [max 10×1 p]

Consider the `numpy.array` A below:

```
A = np.array([[44, 35, 72, 55, 15, 69],
              [23, 42, 82, 74, 11, 61],
              [77, 50, 20, 54,  7, 49]])
```

i. What is the dimensionality of A?

ii. What is the shape?

iii. How can one retrieve `[23, 42, 82, 74, 11, 61]` from A?

iv. ... `[72, 82, 20]`?

v. ... `[11]`?

vi. ... `[42, 82, 50, 20]`?

vii. How can we permute A into the following array?

```
array([44, 35, 72, 55, 15, 69, 23, 42, 82,
       74, 11, 61, 77, 50, 20, 54,  7, 49])
```

viii. How can we permute A into the following array?

```
array([[44, 35],
       [72, 55],
       [15, 69],
       [23, 42],
       [82, 74],
       [11, 61],
       [77, 50],
       [20, 54],
       [ 7, 49]])
```

ix. How can one obtain the mean of A and subtract it from every element, so that we're left with the following array?

```
array([[ -2.66666667, -11.66666667, 25.33333333, 8.33333333,
         -31.66666667,  22.33333333],
        [-23.66666667,  -4.66666667, 35.33333333, 27.33333333,
         -35.66666667,  14.33333333],
        [ 30.33333333,  3.33333333, -26.66666667, 7.33333333,
         -39.66666667,   2.33333333]])
```

x. Consider a new array B below. How can one square every element of A, obtain the product of A^2B and sum all resulting values together, so that one is left with a single value 106584?

```
B = array([[0, 0, 1],
           [0, 0, 1],
           [0, 1, 1],
           [1, 1, 1],
           [1, 1, 1],
           [1, 1, 1]])
```

Question 7 [max 10 p]

Consider the toy dataset provided below, which is taken from SemEval-2018 Task 1: Affect in Tweets. The data consists of text strings representing users' Tweets and their associated valence labels. Here, the labels are integers ranging from -1, which represents a negative emotional state, to 1, which represents a positive emotional state. 0 is neutral. Given the data, provide `python` code for training a `LinearSVC` classifier in `scikit-learn` for valence classification. Your code should include three steps: featurization via `tf-idf`, training, and evaluation. Disregarding imports, your program should not exceed 10 lines. **NOTE:** since this dataset is very small, you should not expect to see a high accuracy.

Toy Dataset

```
X_train = [
    "I was going to say that Rooney is a shadow of his former self \
     but I don't want to offend any shadows. #NTFCvMUFC",
    "Left my phone in Mcds #panic",
    "It's 5:55am. I'm hungry but there is no food. #panic",
    "Charlotte and a friend just climbed to the top of a windmill! \
    Think the view was easy, breezy, beautiful? @PlayHollywoodU",
    "now that I have my future planned out, I feel so much happier \
    #goals #life  #igotthis #yay",
    "A #new day to #live and #smile. Hope all the #followers a nice \
    #night or #day. :D",
    "the ending of how I met your mother is dreadful",
    "#Sports Top seed Johnson chases double delight at \
     Tour Championship",
    "Howard Webb always held up as knowing the answers. No red. \
     Sutton and Craigan raging! Hahaha",
    "Well that was exhilarating. I didn't know you could have \
     goosebumps for 2 hrs 5m straight."
]

Y_train = [-1, -1, -1, 1, 1, 1, -1, 0, 0, 1]

X_test = [
    "The current run of superman continues to be a delight! \
     You can feel the love poured into it by everone on \
     the creative team #superman",
    "Turns out 'it' wasn't even anything to be concerned \
     about at all. Im happy and a bit frustrated it took \
     so long to get this answer.",
    "Don't join @BTCare they put the phone down on you, \
    talk over you and are rude. Taking money out of my \
    acc willynilly! #fuming",
    "Hey @gmail why can I only see 15 sent emails? \
    Where's the thousands gone?  #panic",
    "Congress attended by midterm liveliness differently \
    constraints: qmiv"
]

Y_test = [1, 0, -1, -1, 0]
```