EECE 230 Introduction to Computation and Programming, Sections 4, 5, 6, and 7
Final Exam

May 11, 2019

- The duration of this exam is 2 hours and 55 minutes. Keep in mind that you need around 10 minutes at the end of the duration of the exam to submit your answers. It is your responsibility to make sure your files are correctly submitted.
- The exam consists of 5 problems for 180 points
- You can use all the material in the exam zip file on moodle (lecture slides, source code, programming assignments, and solutions). Once you download the zip file, moodle will be disconnected.
- At the end of the exam, moodle will reopen for exam submission. If you would like to submit your work before the end of the exam, please talk to the proctors for instructions.
- You are asked to submit a single zip file containing your `Python` files (ending with `.py` extension). Failure to do so may lead to a failing grade on the exam. It is your responsibility to make sure your files are correctly submitted.
- You are **NOT** allowed to use the **web**. You are not allowed to use **USB's** or files previously stored on your machine.
- If you get caught violating the above rules or if you communicate with a person other than the exam proctors during the exam, you will immediately get zero and you will be referred to the appropriate disciplinary committee.
- Cell phones and any other unauthorized electronic devices are absolutely not allowed in the exam rooms. They should be turned off and put away.
- The problems are of varying difficulty. Below is a rough ordering estimate of the problems in order of increasing difficulty.
  - Level 0 (10 points): Nonefficient solution of Problem 3
  - Level 1 (103 points): Problem 2 and nonefficient solutions of Problems 1 and 4
  - Level 2 (52 points): Efficient solutions of Problems 1, 3, and 4 and nonefficient solution of Problem 5
  - Level 3 (15 points): Problem 5
- Detailed comments are worth partial credit.
- Plan your time wisely. Do not spend too much time on any one problem. Read through all of them first and attack them in the order that allows you to make the most progress.
- Good luck!

Problem 1 (40 points). Number of distinct elements

Implement the function `numberOfDistinct(L)`, which given a list of numbers L, returns the number of *distinct* elements in L. That is, it counts the number of elements in L disregarding repetition. For example, $L = [2, 1, 2]$ has only two distinct elements: 1 and 2.

Any correct solution is worth 30 points. For full grade, solve it in $O(n \log n)$ time or $O(n)$ expected time, where n is the length of L.

Test program/output:

```
print(numberOfDistinct([5,1,1,0,5,-3,0]))                    | 4
print(numberOfDistinct([8,0,33,9,10]))                       | 5
print(numberOfDistinct([0.3, 0.1, 0.3, -7.2]))               | 3
print(numberOfDistinct([2,2,2,2,2]))                         | 1
print(numberOfDistinct([]))                                  | 0
```

Submit your solution in a file called Prob1.py. Inside the file, include your name and ID number (as comments).

Problem 2 (55 points). Classes

a) **Netflux user class (40 points).** In this problem, you will design an abstract data type for a Netflux user called `NetUser`.

A `NetUser` has three data attributes: a string `name`, a string `email`, and a dictionary `movies`. The dictionary `movies` has movie titles (strings) as keys, and integer ratings as values. Moreover, `NetUser` has the following method attributes:

- `__init__`, which takes `name` and `email` as input arguments. This method should return the exception "Bad Input!" if either `name` or `email` is not a string. It should also initialize the movies dictionary to empty dictionary.
- `addMovie`, which takes a string `title` (representing a movie title) as input and adds it to the dictionary `movies` with a corresponding value of 0.
- `rateMovie`, which takes a string `title` (representing a movie title) and a positive integer `score` (representing the movie rating) as input. It should raise an assertion error if `title` is not in `movies` or if `score` is not a positive integer. Otherwise, it should update the dictionary `movies` so that the value corresponding to `title` is changed to `score`.
- `commonMovies`, which takes as input a NetUser `other`. It should return a list of movies that are common to both users. If the type of `other` is not `NetUser`, it should raise an assertion error.
- `__str__`, which casts `NetUser` as a string in the following form:
  Name: ...
  E-mail: ...
  Number of movies in dictionary: ...
  It's movie time because EECE 230 (which is fun and easy) is over!

Any correct solution is worth full credit.

Test program:

```
IB=NetUser("IbnBalluta The Explorer","ite@sydney.ude.au")
IB.addMovie("The Mask of Zorro")
IB.rateMovie("The Mask of Zorro",10)
IB.addMovie("The 100 year-old man who climbed out of the window and disappeared")
IB.rateMovie("The 100 year-old man who climbed out of the window and disappeared",9)
IB.addMovie("The Matrix")
IB.addMovie("Amelie")

MS=NetUser("Majhoul Al-Shoubasi","mas@gmail.moc")
MS.addMovie("The Matrix")
MS.rateMovie("The Matrix",10)
```

```
MS.addMovie("The Godfather")
MS.addMovie("Howl's Moving Castle")
print(IB,"\n")
print(MS,"\n")
print(MS.commmonMovies(IB))
```

Output:

```
Name: IbnBalluta The Explorer
E-mail: ite@sydney.ude.au
Number of movies in dictionary: 4
It's movie time because EECE 230 (which is fun and easy) is over!

Name: Majhoul Al-Shoubasi
E-mail: mas@gmail.moc
Number of movies in dictionary: 3
It's movie time because EECE 230 (which is fun and easy) is over!

['The Matrix']
```

Submit your solution in a file called Prob2a.py. Inside the file, include your name and ID number (as comments).

b) **Build star graph (15 points).** In this part, you need the file **graph.py** we used in Programming Assignment 11. For convenience, it is also available in the compressed folder `finalFiles.zip`.

Write a function `buildStarGraph(n)`, which given an integer n, returns the undirected star graph with n nodes labeled $0, 1, 2, \ldots, n-1$ (as integers) and centered at node 0, i.e., node 0 is connected to nodes $1, 2, \ldots, n$. See the below examples. Assume that $n \geq 1$, and generate an assertion error if $n \leq 0$.

Any correct solution is worth full credit.

Needed modules:

```
from graph import UndirectedGraph
import matplotlib.pyplot as plt
```

Test program:

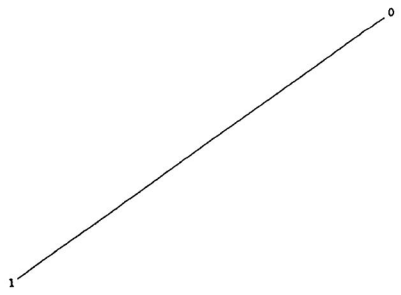
```
for n in (1,2,6):
    G= buildStarGraph(n)
    print(G)
    plt.figure(n)
    plt.clf()
    G.draw()
```

Output 1/3:

```
0 :
```

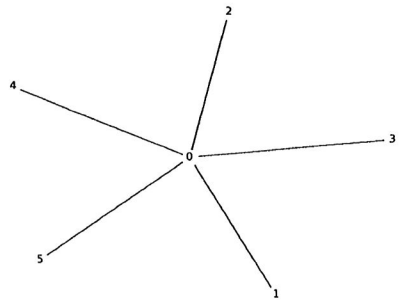

Output 2/3:

```
0 : 1
1 : 0
```



Output 3/3:

```
0 : 1,2,3,4,5
1 : 0
2 : 0
3 : 0
4 : 0
5 : 0
```



Submit your solution in a file called Prob2b.py. Inside the file, include your name and ID number (as comments).

Problem 3 (30 points). Sorting a unimodal list

Recall from Problem 3 of Programming Assignment 3 the definition of a unimodal list:

A non-empty length-n list L is called unimodal if there exists an integer m, $0 \leq m \leq n-1$, such that $L[i-1] < L[i]$ for $i = 1, \ldots, m$ and $L[i-1] > L[i]$ for $i = m+1, \ldots, n-1$. We call the index m the mode of A.

Examples:

- The list `L = [1,2,4,7,11,10,8,4,-9]` is unimodal and its mode is 4 since:
  - $1 < 2 < 4 < 7 < 11$
  - The index of 11 in L is 4
  - $11 > 10 > 8 > 4 > -9$
- The list `L = [1,2,5,20]` is unimodal and its mode is 3 since it is sorted in increasing order and the index of last element is 3.
- The list `L = [50,2,1]` is unimodal and its mode is 0 since it is sorted in decreasing order and the index of first element is 0.
- The list `L = [1]` is unimodal and its mode is 0

Write a function `sortUnimodalList(L)`, which given a unimodal list L, reorders L by sorting it in non-decreasing order. Note that in this problem you are not asked to check if the input list L is unimodal and your algorithm is not supposed to work properly if it is not unimodal.

Any correct solution is worth 10 points. To get full grade, sort L efficiently by taking advantage of the fact that L is not an arbitrary list: aim for $O(n)$ time, where n is the length of L.

You may use any code from class or the assignments' solutions available in the exam folder.

Test program/output:

```
L = [1]
print("L       :", L)
sortUnimodalList(L)
print("L sorted:", L,"\n")
L = [1,2]
print("L       :", L)
sortUnimodalList(L)
print("L sorted:", L,"\n")
L = [1,2,4,7,11,10,8,4,-9]
print("L       :", L)
sortUnimodalList(L)
print("L sorted:", L,"\n")
L = [1,2,5,20]
print("L       :", L)
sortUnimodalList(L)
print("L sorted:", L,"\n")
L = [20,5,2,1]
print("L       :", L)
sortUnimodalList(L)
print("L sorted:", L,"\n")
```

```
L       : [1]
L sorted: [1]

L       : [1, 2]
L sorted: [1, 2]

L       : [1, 2, 4, 7, 11, 10, 8, 4, -9]
L sorted: [-9, 1, 2, 4, 4, 7, 8, 10, 11]

L       : [1, 2, 5, 20]
L sorted: [1, 2, 5, 20]

L       : [20, 5, 2, 1]
L sorted: [1, 2, 5, 20]
```

Submit your solution in a file called Prob3.py. Inside the file, include your name and ID number (as comments).

Problem 4 (30 points). Cumulative matrix

Implement the function `cumulativeMatrix(M)`, which given an $m \times n$ matrix M, returns an $m \times n$ matrix C as follows. For $0 \leq i < m$ and $0 \leq j < n$, $C[i][j]$ is the sum of all the elements in the submatrix of M consisting of the first $(i+1)$ rows and $(j+1)$ columns. That is, $C[i][j] = \sum_{k=0}^{i} \sum_{\ell=0}^{j} M[k][\ell]$. For example, if

$$M = \begin{bmatrix} \underline{1} & \underline{5} & \underline{3} & 4 \\ \underline{-4} & \underline{2} & \underline{10} & 0 \\ 9 & 81 & 7 & 7 \end{bmatrix},$$

then $C[1][2]$ is the sum of the elements of the submatrix of C consisting of the first two rows and first three columns. That is, $C[1][2]$ is the sum of the underlined elements, i.e., $C[1][2] = 17$.

Any correct solution is worth 18 points. For full grade, solve it in $O(mn)$ time.

Test program:

```
import numpy as np
M1 = [[1,2],[3,4]]
M2 = [[5,-2,1]]
M3 = [[1],[10],[3]]
M4 = [[1,5,3,4],[-4,2,10,0],[9,81,7,7]]
for M in (M1,M2,M3,M4):
    print("M")
    print(np.matrix(M))
    print("C")
    print(np.matrix(cumulativeMatrix(M)),"\n")
```

Output:

```
M
[[1 2]
 [3 4]]
C
[[ 1  3]
 [ 4 10]]

M
[[ 5 -2  1]]
C
[[5 3 4]]

M
[[ 1]
 [10]
 [ 3]]
C
[[ 1]
 [11]
 [14]]

M
[[ 1  5  3  4]
 [-4  2 10  0]
 [ 9 81  7  7]]
C
[[  1   6   9  13]
 [ -3   4  17  21]
 [  6  94 114 125]]
```

Submit your solution in a file called Prob4.py. Inside the file, include your name and ID number (as comments).

Problem 5 (25 points). Generate abc-strings without identical consecutive characters

In this problem, an abc-string means a string each character of which is 'a', 'b', or 'c'. We are interested in abc-strings without identical consecutive characters. That is, any two consecutive characters in the string should be different. For instance, 'aba' and 'abcba' do not have identical consecutive characters, but 'baac' and 'bbba' have identical consecutive characters (underlined)

Write a recursive function `genStrings(n)`, which given integer n, returns a list `L` consisting of all length-n abc-strings without identical consecutive characters. See the below examples. If $n < 0$, the function should return the empty list.

Any correct solution is worth 10 points. For full grade, solve it efficiently: aim for $O(nN)$ time, where N is the number of elements in the list L returned by the function.

(Hint: Recursion. Define in `genStrings` a recursive function with appropriate parameters)

Test program:

```
for n in range(6):
    print("genStrings("+str(n)+")")
    print(genStrings(n))
```

Output:

```
genStrings(0)
['']
genStrings(1)
['a', 'b', 'c']
genStrings(2)
['ab', 'ac', 'ba', 'bc', 'ca', 'cb']
genStrings(3)
['aba', 'abc', 'aca', 'acb', 'bab', 'bac', 'bca', 'bcb', 'cab', 'cac', 'cba', 'cbc']
genStrings(4)
['abab', 'abac', 'abca', 'abcb', 'acab', 'acac', 'acba', 'acbc', 'baba', 'babc', 'baca', 'bacb',
 'bcab', 'bcac', 'bcba', 'bcbc', 'caba', 'cabc', 'caca', 'cacb', 'cbab', 'cbac', 'cbca', 'cbcb']
genStrings(5)
['ababa', 'ababc', 'abaca', 'abacb', 'abcab', 'abcac', 'abcba', 'abcbc', 'acaba', 'acabc', 'acaca',
  'acacb', 'acbab', 'acbac', 'acbca', 'acbcb', 'babab', 'babac', 'babca', 'babcb', 'bacab', 'bacac',
  'bacba', 'bacbc', 'bcaba', 'bcabc', 'bcaca', 'bcacb', 'bcbab', 'bcbac', 'bcbca', 'bcbcb', 'cabab',
  'cabac', 'cabca', 'cabcb', 'cacab', 'cacac', 'cacba', 'cacbc', 'cbaba', 'cbabc', 'cbaca', 'cbacb',
  'cbcab', 'cbcac', 'cbcba', 'cbcbc']
```

Submit your solution in a file called Prob5.py. Inside the file, include your name and ID number (as comments).