CS331- Introduction to Artificial Intelligence

Instructor: Yasir Mehmood

Assignment – 1

Prepared by Syed Suleman Ahmad, Mishaal Kazmi, Muhammad Ammar

Submission Deadline: 16th February

(There will be No Extensions)

Honor Code

You are to do this assignment by yourself. All of the submitted code should be your creation and a result of your own thought process. No cooperation is allowed under any circumstances. Any help outside of the course staff is prohibited. You are also given the responsibility of reporting any cooperation incidences to the course staff. All code will be checked for similarities, and cases will be promptly forwarded to the Disciplinary Committee for action.

About Python:

Python is not a difficult language to learn as you will see while attempting this assignment. Following are some links that will help you get started:

https://www.programiz.com/python-programming
https://www.tutorialspoint.com/python/
http://thepythonguru.com/
http://www.w3resource.com/python/python-tutorial.php/

Important Instructions / Submission Guidelines:

- You will need to download and setup Python. It is fairly easy, in case of any trouble contact the TAs. We will be using Python 2.7 version.
- You should submit a zip folder with all your files in it. Name the folder "CS331 - Assignment1-Your_roll_number". Example: CS331-Assignment1-2010xxxx.zip
- Include **one** file for each part in the zip folder. Each filename mentioned in the question must be strictly followed and only these files must be submitted.
- Submit this folder on LMS BEFORE the deadline. **No late submissions will be accepted.**
- We will be trying to automate the checking process so please **follow the guidelines in each problem** to avoid any inconvenience. On top of that, it does not mean that we won't check your code to understand your logic if you have gone wrong somewhere.
- If your code encounters a runtime error or does not run at all then a major portion of your grade will be cut off.

Please read the handout carefully and follow the instructions.

Problem 1: (Filename: Palindrome.py)

Palindrome: A string or sequence that reads the same backwards as forwards, e.g. *madam, 2002*

Your task is to write a function which finds the **length and substring** of the longest palindrome, in a given string.

Example: "AABBCBBAC" -> Longest Palindrome: "ABBCBBA" of length 7.

Input Format: Take input from User.

Output Format: Print on console.

Function Prototype: LongestPalindrome(text)

Problem 2: (Filename: StackTheQueue.py)

Your task is to implement a Stack data structure using Queues. A file named util.py has been provided to you, and it contains the Stack class structure. You may use the Stack class definition provided to you as reference and implement a Queue class **in the same util.py file.** Please stay consistent with the class function names provided in util.py

After completing the Queue class, you need to edit 'StackTheQueue.py' (provided to you) to implement 'StackTheQueue' class so that it acts as a stack. There is a small test case in main function as well.

Hint: You can use two Queues to implement a stack.

Stack: A structure which observes LIFO (Last in First Out Policy). In other words, the elements are pushed in later are emitted first.

Queue: A structure which observes FIFO (First in First out Policy) policy.

Task skeleton has been provided; you just need to edit the functions in both files.

For the following two questions, please write the logic behind your solutions in form of comments for these problems at the top of your code. You will only get partial marks without an explanation. You will be required to make an input file to read the problem data for each question, BUT ONLY SUBMIT THE CODE FILE NOT THE INPUT FILES. Google file reading its quite simple in python.

Problem 3: (Filename AnnualPlay.py)

It is opening night at the LUMS Annual Play, and your friend is in the lead role. You will not be in the audience, but you want to make sure he receives a standing ovation -- with every audience member standing up and clapping their hands for him.

Initially, the entire audience is seated. Everyone in the audience has a hesitation level. An audience member with hesitation level $\mathbf{H}_i$ will wait until at least $\mathbf{H}_i$ other audience members have already stood up to clap, and if so, he will immediately stand up and clap. If $\mathbf{H}_i = 0$, then the audience member will always stand up and clap immediately, regardless of what anyone else does. For example, an audience member with $\mathbf{H}_i = 2$ will be seated at the beginning, but will stand up to clap later after he sees at least two other people standing and clapping.

You know the hesitation level of everyone in the audience, and you are prepared to invite some additional friends to be in the audience to ensure that everyone in the crowd stands up and claps in the end. Each of these friends may have any hesitation value that you wish, not necessarily the same. What is the **minimum** number of friends that you need to invite to guarantee a standing ovation?

Input Format: Make/Read ovation.txt. The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each consists of one line with $\mathbf{H}_{max}$, the maximum hesitation level of the most hesitant person in the audience, followed by a string of $\mathbf{H}_{max} + \mathbf{1}$ single digits (assume them to be numbers or ints). The **i**th digit of this string (counting starting from 0) represents how many people in the audience have hesitation level **i**. For example, the string "409" would mean that these were four audience members with $\mathbf{H}_i = 0$ and nine audience members with $\mathbf{H}_i = 2$ (and none with $\mathbf{H}_i = 1$ or any other value). Note that this will initially always be between 0 and 9 people with each hesitation level.

The string will never end in a 0. Note that this implies that this will always be at least one person in the audience.

$1 \leq \mathbf{T} \leq 10$. (test cases)
$0 \leq \mathbf{H}_{max} \leq 100$

Output Format: For each test case, output on console one line containing "Case x: y", while x is the test case number (starting from 1) and y is the minimum number of friends you must invite**.**

Sample (Function prototype: standingOvation(H_{max}, str)

Input	**Output**
4	
4 11111	Case 1: 0
1 09	Case 2: 1
5 110011	Case 3: 2
0 1	Case 4: 0

Sample Explanation:

In **Case 1**, 4 is $\mathbf{H}_{max}$ (The person in the audience with max hesitation level) followed by $\mathbf{H}_{max}$ + 1 digits i.e 5 digits. So, as explained above there is 1 person with hesitation level $\mathbf{H}_i = 0$, 1 with $\mathbf{H}_i = 1$, one person

with H_i = 2 and so on. Read the paragraph under the Input heading again if you don't yet understand this.

The audience will eventually produce a standing ovation on its own, without you needing to add anyone -- first the audience member with $H_i = 0$ will stand up, then the audience member with $H_i = 1$ will stand up, then then the audience member with $H_i = 2$ will stand up etc.

In **Case 2**, a friend with $H_i = 0$ must be invited, but that is enough to get the entire audience to stand up.

In **Case 3,** one optimal solution is to add two audience members with Hi = 2.

Problem 4: (Filename Sheep.py)

Tonton the sheep has devised a strategy that helps her fall asleep faster. First, she picks a number **N**. Then she starts naming **N** , 2 × **N** , 3 × **N** , and so on. Whenever she names a number, she thinks about all of the digits in that number. She keeps track of which digits (0, 1, 2, 3, 4, 5, 6, 7, 8, and 9) she has seen at least once so far as part of any number she has named. Once she has seen each of the ten digits at least once, she will fall asleep.

Tonton must start with **N** and must always name (*i* + 1) × **N** directly after *i* × **N**. For example, suppose that Tonton picks **N** = 1692. She would count as follows:

- **N** = 1692. Now she has seen the digits 1, 2, 6, and 9.
- 2 **N** = 3384. Now she has seen the digits 1, 2, 3, 4, 6, 8, and 9.
- 3 **N** = 5076. Now she has seen all ten digits, and falls asleep.

What is the last number that she will name before falling asleep? If she will count forever, print INSOMNIA instead.

Input Format: Make/Read sheep.txt. The first line of the input gives the number of test cases, T. T test cases follow. Each consists of one line with a single integer N, the number Tonton has chosen.

Output Format: Print on Console. For each test case, output one line containing Case x: y , where x is the test case number (starting from 1) and y is the last number that Tonton will name before falling asleep, according to the rules described in the statement.

Function Prototype: Sheep(N)

Sample:

Input	Output
5	
0	Case #1: INSOMNIA
1	Case #2: 10
2	Case #3: 90
11	Case #4: 110
1692	Case #5: 5076

Problem 5: (Filename NUMPY.py)

This is a simple exercise related to the Python NumPy Library. Just like in C++ you can 'include' different libraries in your program; in python you can 'import' modules. NumPy is a linear algebra module which will be really helpful in your future assignments. We will use this library to create and work with matrices/NumPy arrays.

Please follow the instructions in the link below to install and setup numpy.
https://docs.scipy.org/doc/numpy-1.13.0/user/setting-up.html

To import NumPy in your program write at the top:

import numpy as np

The 'as np' is added so that reference to methods in this library becomes less wordy (instead of numpy.array we can do np.array)

Part 1:

```
#input dataset
inputData = [ [0, 0, 1],
              [0, 1, 1],
              [1, 0, 1],
              [1, 1, 1] ]

print("Input as single line list (we do not want the input to look like this):")
print(inputData)
```

After executing the above code we get the output as:

[[0, 0, 1], [0, 1, 1], [1, 0, 1], [1, 1, 1]]

You can see that the output is a list where as we require that our output should be in the form of a matrix/vector as shown below: [4 rows 3 columns in this case]

```
[[0 0 1]
[0 1 1]
[1 0 1]
[1 1 1]]
```

So your task in this part is just to print in a matrix format using numpy matrices. The purpose behind this exercise is for you to learn how to use the python numpy library to make numpy arrays (matrices or vectors basically) which is the key format in which our data is supposed to look like for most problems in AI where we have a set of columns as input and a set of column(s) as output. Make use of the python numpy library to achieve this.

Useful way to start is understand the problem and try researching on how to change lists to numpy arrays using numpy in python.
Help: http://www.scipy-lectures.org/intro/numpy/array_object.html

Part 2:

For this part u will learn to generate random matrices/vectors using numpy. This will prove useful later when we need to generate random weights for our neural networks (more on this later in the course).

For now your task is only to generate a random 3x1 vector/matrix (3 rows and 1 column of random numbers) and 5x2 vector/matrix (5 rows, 2 columns randomly populated) using the numpy library.

Research on how to generate random matrix or arrays in numpy. Please note we are using the words vector, matrix and array interchangeably as they mean mostly the same thing in numpy.

Print both the matrices on the console.

Part 3:

inputData = [[0, 0, 1],
[0, 1, 0],
[1, 0, 0],
[1, 1, 1]]

Your task in this part is to separate the above matrix into two matrixes of size 4x2 and 4x1. You will make two numpy arrays, one will store the first two columns of the matrix and the other will only store the last column of the above matrix.
This exercise will help you learn how to index and slice data using numpy because we wll need this later on when we want to split our data into input and output.

Print both arrays on console.

Help: https://docs.scipy.org/doc/numpy-1.13.0/user/quickstart.html

Part 4:

Make two random numpy arrays of size 5 x 4 and 4 x 1.
Your task is to print the 'matrix product' (of size 5 x 1) of both the randomly generated matrixes.

Use the link above for help.

These are very small exercises but you will appreciate them in the later part of the course as mentioned ☺