SSN College of Engineering

Department of Information Technology

UIT2201 — Programming and Data Structures

2021 – 2022

Exercise — 08

May 25, 2023

- This homework is due by 10PM on May 30, 2023
- Grace period may be given up to midnight of May 30, 2023
- You can upload only one ZIP file
- The naming convention is "<Your first name (first letter capital and all the other letters small)>-UIT2201-ex-08.zip"
- Judicious use of Python features and standard modules, version control using 'git', adhering to Python coding standards are expected
- You are expected to use PSP0.1 process for all the code that you write!

The purpose of this exercise is to understand the design and implementation of ADT List using low-level arrays.

1. Provide an implementation of List ADT, using dynamically managed array with the following methods.
    - Constructor, which takes an argument called 'val' and creates a new list. If the argument 'val' is an integer, you should create an array of initial capacity 'val'. If the argument 'val' is a sequence (such as list, tuple, str, etc.) then your constructor should create an array of double the size of the sequence 'val' and initialize the newly created list with the elements of the sequence 'val'.
    - A string method (*__str__*) to convert this list object into a string.
    - A length method (*__len__*) that returns the size of this list object.
    - A *__getitem__* method to return the object stored at the given index.
    - A *__setitem__* method to update the object stored at the given index.
    - An append method (*append*), that adds the given object at the end if this list. Note that the underlying array should be dynamically managed and there should not be any capacity limit.
    - An insert method (*insert*) to insert the given object at the given index. The size of this list increases by 1, and there should not be any capacity limit. This list is mutated and no new list is created.

- A delete method (*delete*()) to delete the object at the given index. This list is mutated and the size reduces by 1. If the size of the list falls below 25% of the current capacity, then resize the underlying array to half of its capacity.

2. Implement two different versions of the following list operations. First option is to implement them as basic methods (inside the class) and the second option is to define them as general functions (outside the class). Discuss the merits and demerits of both. Note that the methods have access to the fields of the data structure, while general functions can only use the public methods declared within the class (as given in Question 1).

   - An extend utility (*extend*()) to extend the first list ('self' if implemented as a method) with the elements of the second list.
   - *contains*() functionality that returns 'True' is the given object is present in the list, 'False' otherwise. When implemented as a method, it should be named as__*contains*_().
   - *index*() functionality that returns the index of the first occurrence of the given object in the list. An exception should be raised if the object is not present in the list.
   - *count*() functionality that the returns the count of occurrences of the given object in the list.

3. Let us perform an empirical analysis to understand the time complexity of the 'append()' method. Write a function (takes an integer n as an argument) that creates an empty list and append n random objects to that list. Your function should record the time taken T for these n appends, and return the average time T/n.

   Run the experiment for different (very large) values of n and note down the average time taken per 'append()' operation. How does this average time increase as n increases? Comment on your observation.