National University of Sciences & Technology (NUST)
School of Natural Sciences (SNS)
Department of Mathematics

CS250: Data Structures and Algorithm

Class: BS Mathematics - 2021

Lab 06:

Linked Lists based implementation of Lists ADT in Python

Date: 18th October, 2023

Time: 10:00 am - 01:00 pm

Instructor: Fauzia Ehsan

National University of Sciences & Technology (NUST)
School of Natural Sciences (SNS)
Department of Mathematics

Lab 06: Implementation of Lists ADT using Linear data structures in Python

Introduction
Students have learned the fundamental concepts of **linked lists** in the lectures. This lab will introduce students with the practical implementation of a linked list and different operations that can be performed on a linked list.

Objectives
Objective of this lab is to get familiar with List ADT and its implementation using **singly linked list** and implement them in Python.

Tools/Software Requirement
Python 3/ PyCharm IDE / VS Code

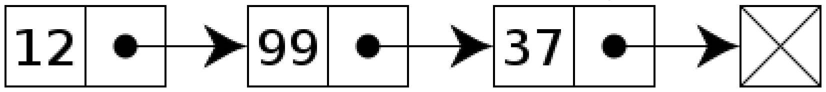
Contents

1. Linked List:

Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array.

Singly Linked List

A Linked List is a data structure consisting of a group of nodes which together represent a sequence. Under the simplest form, each node is composed of two parts i.e. data part and a reference part (also known as, a link) to the next node in the sequence. This structure allows efficient insertion or removal of elements from any position in the sequence.



Operations:

The basic operations consist of

- ***Creating*** the list.
- ***Initialize*** pointers to NULL.
- ***Inserting*** nodes at the beginning, last and from a specific location.
- ***Deletion*** of nodes from beginning, last and from a specific location.
- ***Traversing*** the list.
- ***Destroying*** the list.

Solution

Task 1a: Create a Singly linked list in Python along with all of its operations mentioned above.

Test your code in main.

Task1 Code

Task 1 Output Screenshot:

Solution

Task 1b: Based upon the linked list created above, write methods to compare two lists:

Implement the __eq__ and __ne__ methods. For these functions, equality should be defined as follows: both lists have the same number of elements, and each pair of corresponding elements in the list are also equal (as defined by the == operator). You should implement only one of these operators from scratch; the other should delegate to the first.

Also, Implement the `remove` method. You will need to traverse the list looking for the item to remove, keeping track of the predecessor node so that you can reconnect the two separate parts of the list after removal. If the item is not found in the list, the method should raise a `ValueError`.

Task1b Code:

Task 1b Output Screenshot:

Task2.

1. **Based on linked list structure above, implement following functions in Python program:**
   a. **Given an unsorted singly linked list, remove all duplicates from the linked list.**
   b. **Implement the function `count_50s()` as a new operation for the linked list structure.**

(Use the usual node definition with member variables called data and pointer to next node)

```
 // Precondition: head_ptr is the head pointer of a linked list.
  // The list might be empty or it might be non-empty.
  // Postcondition: The return value is the number of occurrences
  // of 50 in the data field of a node on the linked list.
  // The list itself is unchanged.
```

   c. **Write a method *`removeEvens()`* that removes the values in even-numbered indexes from a list, returning a new list containing those values in their original order. For example, if a variable list1 stores these values:**

```
list1: [8, 13, 17, 4, 9, 12, 98, 41, 7, 23, 0, 92]
```

And the following call is made:

`list2 = SLL()` #list2 is also of link list type

`list2 = list1.removeEvens()`

After the call, `list1` and `list2` should store the following values:

list1: [13, 4, 12, 41, 23, 92]

list2: [8, 17, 9, 98, 7, 0]

Notice that the values stored in `list2` are the values that were originally in even-valued positions (index 0, 2, 4, etc.) and that these values appear in the same order as in the original list. Also notice that the values left in list1 also appear in their original relative order. **Use constructor to initialize the linked list.**

Hint: You are not allowed to create any new nodes or to change the values stored in data fields to solve this problem; You must solve it by rearranging the links of the list.

Check your functions by calling them in main and add screen shots of output.

d. Odd-Even Linked List.

Given a singly linked list, write a program to group all odd nodes together followed by the even nodes.
Note here we are talking about the node number and not the value in the nodes.

- The relative order inside both the even and odd groups should remain as it was in the input.

- The first node is considered odd, the second node even, and so on.

Example 1

```
Input: 1->2->3->4->5->NULL
Output: 1->3->5->2->4->NULL
Explanation: The first group is of elements at odd position
(1,3,5) in the linked list and then the ones at the even
position(2,4)
```

```
Example 2.
Input: 2->1->3->5->6->4->7->NULL

Output: 2->3->6->7->1->5->4->NULL

Explanation: The first group is of elements at odd position
(2,3,6,7) in the linked list and then the ones at the even
position(1,5,4)
```

Grade Criteria
This lab is graded. Min marks: 0. Max marks: 30

Tasks Shown during Lab (At least 2 Tasks must be shown to LE during lab working) 10 marks	Modern Tool Usage 5 marks	Lab Ethics 5 Marks	Lab Report and Tasks Final Submission 5 marks	Lab Viva/Quiz 5 Marks
Total Marks: 30	Obtained Marks: _____			

Happy Coding!

Deliverables
Comment your program heavily. Intelligent comments and a clean, readable formatting of your code account for 20% of your grade.
You should submit your codes and report as a compressed zip file. It should contain all files used in the exercises for this lab.
The submitted file should be named cs250_firstname_lastname_lab06.zip