CS102L Data Structures & Algorithms
Lab 3

Department of Computer Science

Dhanani School of Science and Engineering

Habib University

Spring 2024

Contents

Introduction

1.1 Instructions

- This lab will contribute 1% towards your final grade.
- The deadline for submission of this lab is at the end of lab time.
- The lab must be submitted online via CANVAS. You are required to submit a zip file that contains all the *.py* files.
- The zip file should be named as *Lab_03_aa1234.zip* where *aa1234* will be replaced with your student id.
- **Files that don't follow the appropriate naming convention will not be graded.**
- **Your final grade will comprise of both your submission and your lab performance.**

1.2 Marking scheme

This lab will be marked out of 100.

- 50 Marks are for the completion of the lab.
- 50 Marks are for progress and attendance during the lab.

1.3 Lab Objectives

In this lab, you will implement the List Abstract Data Type (ADT) and its methods using Python List.

1.4 Late submission policy

There is no late submission policy for the lab.

1.5 Use of AI

Taking help from any AI-based tools such as ChatGPT is strictly prohibited and will be considered plagiarism.

1.6 Viva

Course staff may call any student for Viva to provide an explanation for their submission.

Prerequisites

2.1 Assert Statements

Assert statements are crucial tools in programming used to verify assumptions and test conditions within code. For instance, in Python, an assert statement is typically as follows:

```
    assert condition, message
```

For example, in a function calculating the square root, in order to ensure that the input value is non-negative, one might add the following assert statement:

```
assert x >= 0, "x should be a non-negative number"
```

If the condition evaluates to False, an AssertionError is raised, highlighting the issue and its corresponding message.

2.2 Pytest

Pytest is a popular testing framework in Python known for its simplicity and power in facilitating efficient and scalable testing for applications and software. In order to use Pytest, one first needs to install it and create test functions prefixed with *test_* in a Python file. To install Pytest, you can write the following command on the terminal:

```
    pip install pytest
```

After installing Pytest, one can execute all the tests by typing the following command on the terminal:

```
    pytest
```

If you want to test any particular file, then you will have to specify the filename as follows:

```
    pytest <filename>.py
```

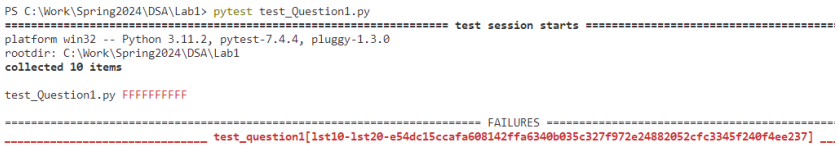
If all the tests pass successfully, then you should receive a similar output.

```
PS C:\Work\Spring2024\DSA\Lab1> pytest test_Question1.py
============================= test session starts ========
platform win32 -- Python 3.11.2, pytest-7.4.4, pluggy-1.3.0
rootdir: C:\Work\Spring2024\DSA\Lab1
collected 10 items

test_Question1.py ..........

============================== 10 passed in 0.06s =========
```

However, if all the test fail, then you should receive a similar output.



Lab Exercises

3.1 Helper functions

3.1.1 Problem Definition

The List ADT is a collection of items where each item is relative to the others. In this question, you will be developing helper functions that you will use in the later questions and in the following labs.

3.1.2 Function Descriptions

The helper functions that you will be developing in this question and their descriptions are as follows:

1. `Initialize(n)`: Creates and returns a new list of size `n`, where all elements are initialized to `None`.
2. `Get(list,index)`: Retrieves the element at the specified `index` in the given list
3. `Set(list,index,value)`: Sets the element at the specified `index` in the given list to the provided `value`.
4. `Size(list)`: Returns the size of the given list.
5. `NumberOfElements(list)`: Returns the number of elements in the given list.
6. `IsEmpty(list)`: Returns `True` if the given list is empty.
7. `IsFull(list)`: Returns `True` if the given list is full.

All these functions should be implemented in the file **Question1.py**, accessible within the Lab 03 module on CANVAS.

3.1.3 Sample

```
>> ListADT = Initialize(5)
>> ListADT
[None,None,None,None,None]
>> Set(ListADT,0,10)
>> ListADT
[10,None,None,None,None]
>> Get(ListADT,0)
10
>> Size(ListADT)
5
>> NumberOfElements(ListADT)
1
>> IsEmpty(ListADT)
False
>> IsFull(ListADT)
False
```

3.1.4 Note

Do not use other modules, help functions, or global variables.

3.1.5 Testing

In order to test your function, type the following command on the terminal:

```
pytest test_Question1.py
```

3.2 Inserting Elements in List ADT

3.2.1 Problem Definition

The processing of a new inserting a new element in List ADT of fixed size is illustrated in the figure below:

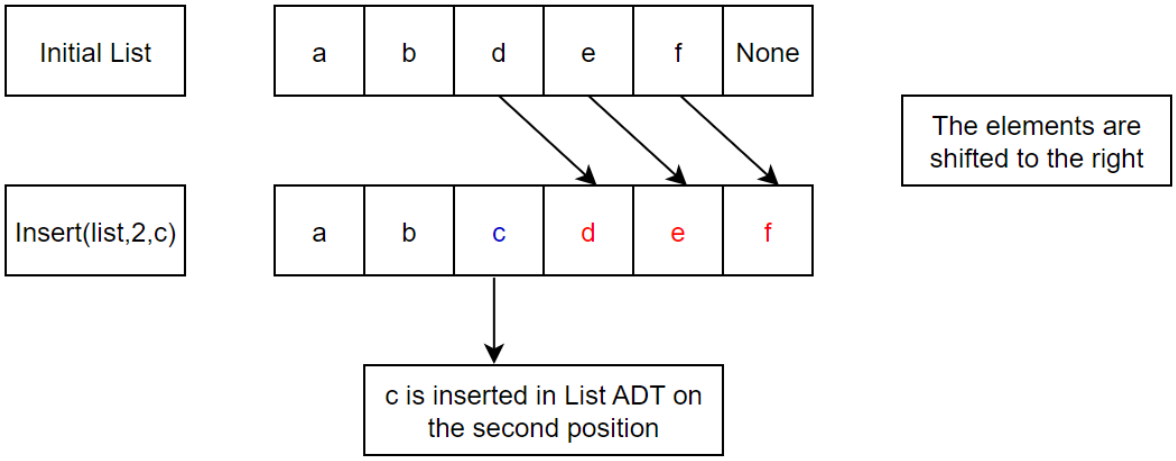


Figure 3.1: Inserting a new element in a List ADT with fixed size

3.2.2 Function Description

Write a function `Insert(list,index,element)` that inserts a new `element` in the given list at the specified `index`. The functions returns one of the following statements depending on the result of the insertion:

- If the list is full, the function will return `"List is full"`
- If the index is invalid, the function will return `"Index is invalid"`
- If the element is inserted successfully, the function will return `"Element inserted successfully"`

This function should be implemented in the file **Question2.py**, accessible within the Lab 03 module on CANVAS.

3.2.3 Function Working

The `Insert` function works as follows:

1. Check if the given list is full.
2. If the list is not full, check if the given index is valid.
3. If the index is valid, shift the elements to the right.
4. Insert the new element at the position specified using index.

3.2.4 Sample Interaction

```
>>ListADT = [None,None,None]
>>Insert(ListADT,0,'a')
"Element inserted successfully"
>>ListADT
["a",None,None]
>>Insert(ListADT,1,'b')
"Element inserted successfully"
>>ListADT
["a","b",None]
>>Insert(ListADT,4,'e')
"Invalid Index"
>>Insert(ListADT,2,'c')
"Element inserted successfully"
>>ListADT
["a","b","c"]
>>Insert(["a","b","c"],3,'d')
"List is full"
```

3.2.5 Testing

In order to test your function, type the following command on the terminal:

```
pytest test_Question2.py
```

3.3 Removing Elements from List ADT

3.3.1 Problem Definition

The removal of a element from a List ADT of fixed size is summarized as follows:

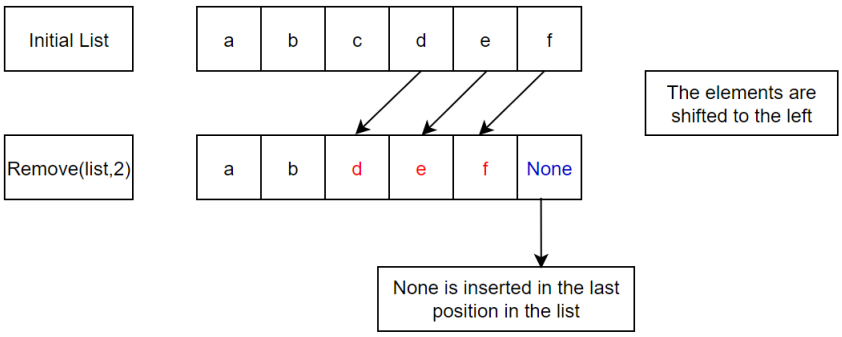


Figure 3.2: Removing an element from a List ADT with fixed size

3.3.2 Function Description

Write a function `Remove(list,index)` that removes a `element` from the given list at the specified `index`. The functions returns one of the following statements depending on the result of the removal:

- If the list is empty, the function will return `"List is empty"`
- If the index is invalid, the function will return `"Index is invalid"`
- If the element is removed successfully, the function will return `"Element removed successfully"`

This function should be implemented in the file **Question3.py**, accessible within the Lab 03 module on CANVAS.

3.3.3 Function Working

The `Remove` function works as follows:

1. Check if the given list is empty.
2. If the list is not empty, check if the given index is valid.
3. If the index is valid, shift the elements to the left.
4. Set the value of the last element equal to `None`

3.3.4 Sample Interaction

```
>> ListADT = [10, 20, 30, 40, 50]
>> Remove(ListADT, 2)
"Element removed successfully"
>>ListADT
[10, 20, 40, 50, None]
>> Remove(ListADT, 0)
"Element removed successfully"
>>ListADT
[20, 40, 50, None,None]
>> Remove(ListADT, 5)
"Invalid Index"
>>Remove([None,None,None,None],0)
"List is empty"
```

3.3.5 Testing

In order to test your function, type the following command on the terminal:

```
pytest test_Question3.py
```

3.4 Insert element at the start of the List

3.4.1 Function Description

Write a function `InsertAtStart(list,element)` that inserts a new element at the start of the given list. The functions returns one of the following statements depending on the result of the insertion:

- If the list is full, the function will return `"List is full"`
- If the element is inserted successfully, the function will return `"Element inserted successfully"`

This function should be implemented in the file **Question4.py**, accessible within the Lab 03 module on CANVAS.

3.4.2 Function Working

The `InsertAtStart` function works as follows:

1. Check if the given list is empty.
2. If the list is not full, shift the elements to the right.
3. Insert the new element at the starting position.

3.4.3 Sample Interaction

```
>> ListADT = [None,None]
>> InsertAtStart(ListADT,'a')
"Element inserted successfully"
>> ListADT
["a",None]
>> InsertAtStart(["a","b"],'c')
"List is full"
```

3.4.4 Testing

In order to test your function, type the following command on the terminal:

```
pytest test_Question4.py
```

3.5 Remove element from the start of the List

3.5.1 Function Description

Write a function `RemoveFromStart(list)` that removes the starting element from the given list. The functions returns one of the following statements depending on the result of the removal:

- If the list is empty, the function will return `"List is empty"`
- If the element is removed successfully, the function will return `"Element removed successfully"`

This function should be implemented in the file **Question5.py**, accessible within the Lab 03 module on CANVAS.

3.5.2 Function Working

The `RemoveFromStart` function works as follows:

1. Check if the given list is empty.
2. If the list is not empty, shift the elements to the left.
3. Set the value of the last element equal to `None`

3.5.3 Sample Interaction

```
>> ListADT = ["a","b"]
>> RemoveFromStart(ListADT)
"Element removed successfully"
>> ListADT
["b",None]
>> >> RemoveFromStart([None,None])
"List is empty"
```

3.5.4 Testing

In order to test your function, type the following command on the terminal:

```
pytest test_Question5.py
```