North Western University, Khulna

Report

Course Code: CSE -2104

Course Title: Data structure Laboratory

Special Thanks to:

Md. Shymon Islam

Lecturer

Department Of CSE

North Western University Khulna, Bangladesh

Developed by:

Shuvro Dobey

Student Id: 20221036010

Miftahul Zannat

Student Id: 20221171010

Ayesha Hossen Erom

Student Id: 20221015010

Department Of CSE

Department Of CSE

North Western University Khulna, Bangladesh

CSE -2104

Table of Contents

Introduction

A simple implementation of a data structure project that involves array and linked list operations using the Tkinter library for creating a graphical user interface (GUI).

- An **array** is a data structure that stores a fixed-size sequence of elements of the same data type in contiguous memory locations.

- A **singly linked** list is a data structure that consists of a sequence of nodes, where each node contains a value and a reference (or link) to the next node in the sequence. It forms a linear collection of elements, where the order is determined by the links between nodes.

Objectives

The project aims to demonstrate basic data structure operations such as insertion, deletion, searching, and updating elements in both arrays and linked lists.

Class & Method Overview:

class Node

func __init__

class LinkedList

func __init__

func add_begin

func add_end

func add_after_x

func add_before_x

func delete_node

func delete_begin

func delete_end

func search_element

func update_element

func display_list

class Ssd

func __init__

func play_sound

func home_interface

func array_interface

func create_array

func array_interface_2

func show_array

func array_insert

func insert_interface

func array_delete

func delete_by_index

func delete_by_index_interface

func delete_by_value

func delete_by_value_interface

func array_search

func search_interface

func array_update

func update_interface

func linkedlist

func add_begin

func add_end

func add_after_x

func add_before_x

func delete_node

func delete_begin

func delete_end

func search_element

func update_element

func display_list

func clear_placeholder

func set_placeholder

func button_click

func clear_entries

Briefing

1. First, we need to import the necessary modules, including **tkinter**, **messagebox**, and **subprocess** for button based sound play.

   Then we define three classes: **Node**, **LinkedList** and **Ssd**

2. **Node class:** Represents a single node in a linked list. Each node contains data and a reference to the next node.

3. **LinkedList class:** Implements the linked list data structure. It includes methods for adding nodes at the beginning or end, adding nodes after or

before a specific node, deleting nodes, searching for elements, updating elements, and displaying the linked list.

4. **Ssd class**: The Ssd class inherits from the LinkedList class and serves as the main class for the GUI application. It creates a window using Tkinter and defines various methods for different operations on arrays and linked lists.

   The Ssd class has an initialization method that sets up the GUI interface and creates an empty array. It also defines methods for different GUI interfaces, such as the home interface, array interface, and linked list interface. Additionally, it includes methods for performing array operations like inserting, deleting, searching, and updating elements.

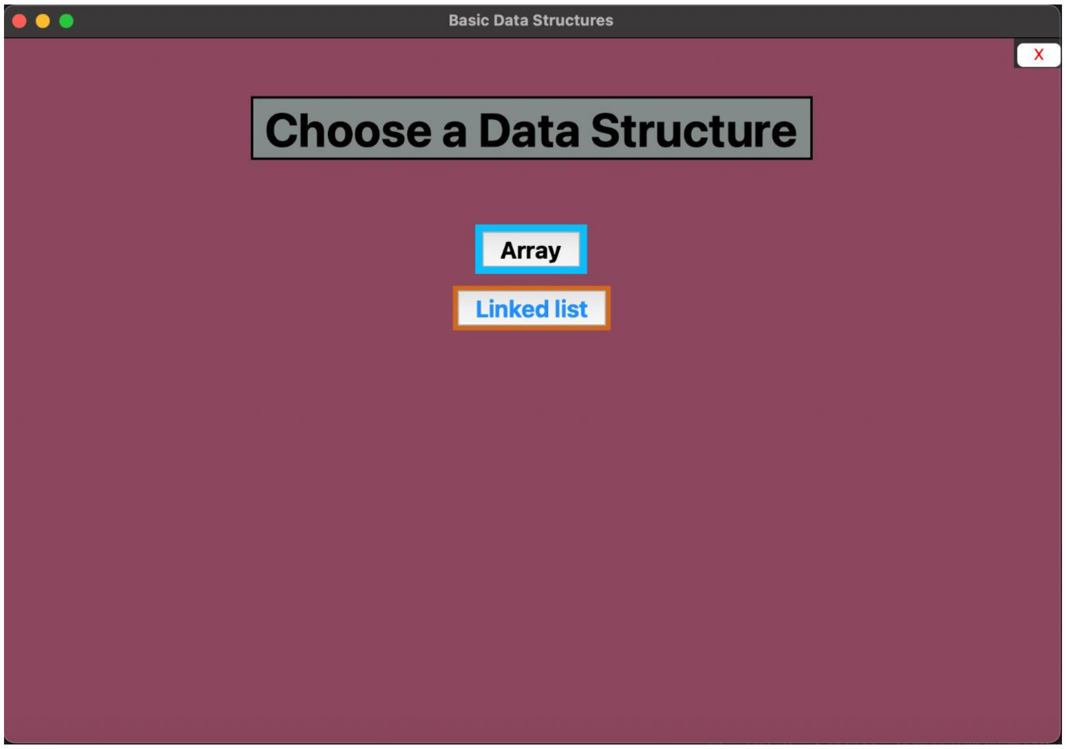
   Tkinter's widgets such as labels, buttons, and entry fields to create the GUI components and handle user interactions. Each GUI interface is created in a separate window using the Toplevel class.

5. Overall, it is a basic framework for a GUI application that allows users to perform operations on arrays and linked lists using a graphical interface.

Description

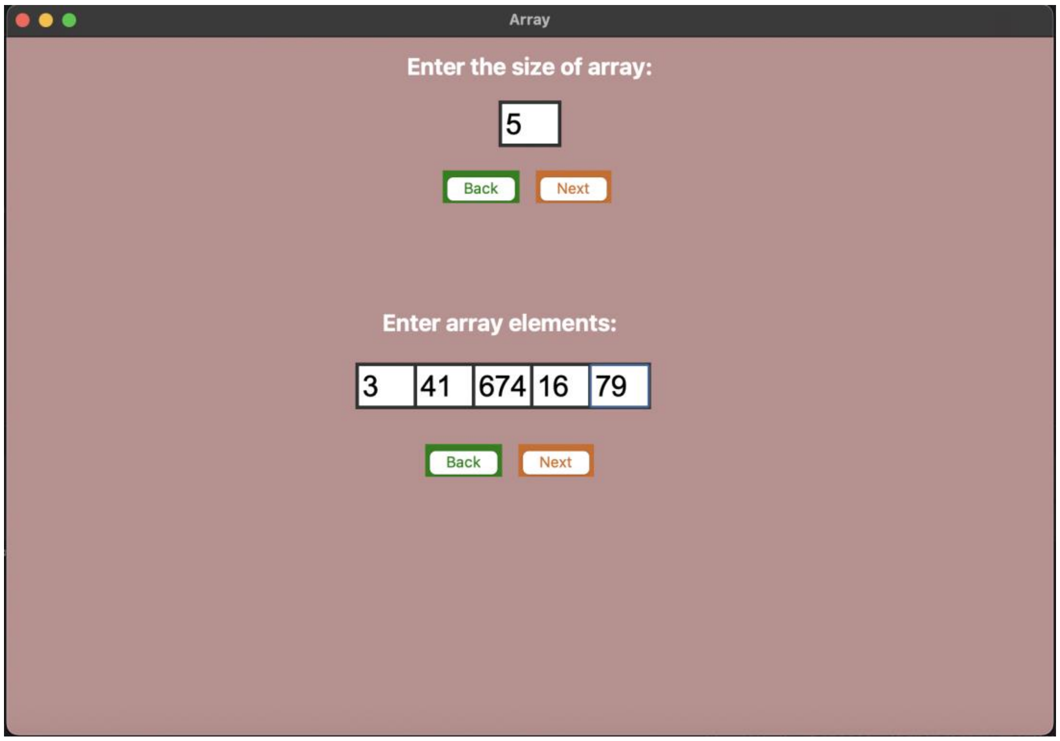
Home page:

This is the home page of the data structure project. There is two buttons. To perform array operation press "*array*" button and, press the "*linked list*" button to perform the linked list operation. And the top-right corner "*X"* is the exit button.



Array:

By clicking the “*array*” button, you go to the array page.

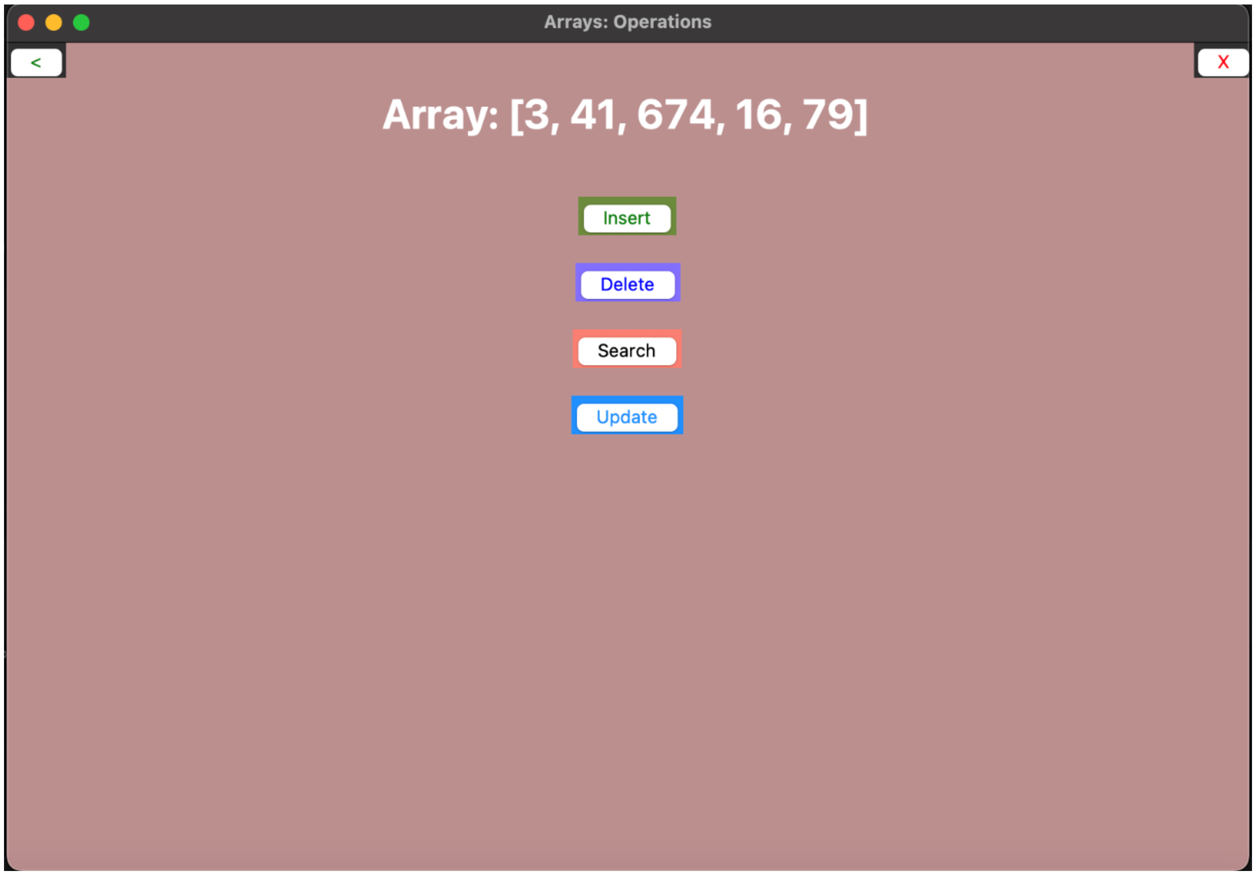


To create an array we should input the array size and then hit "*Next*" button .Now we can see some entry boxes for input the array elements.

So enter the elements in the array. Then click "*Next*" button to create the array. If else you want to go to the home page click "*Back*" button.

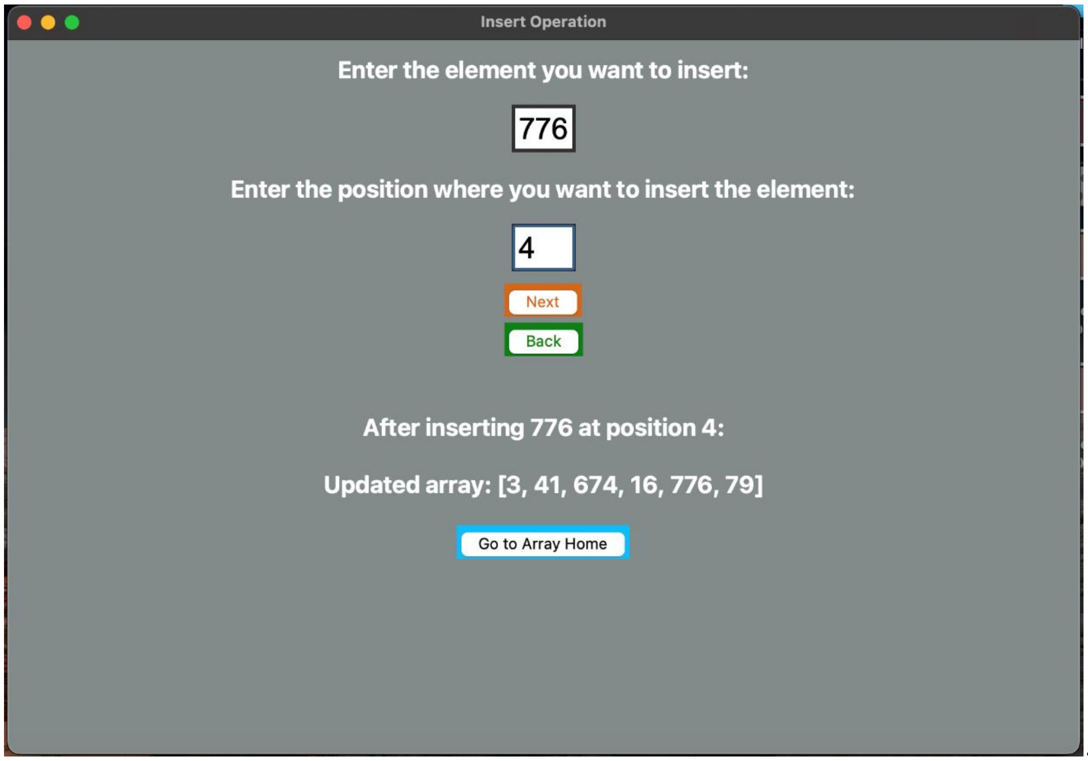
Array Operation:

After creating an array, you will see the array operation page. You can perform the insertion, deletion, search element and update element operations you want to do just press the button.



Insertion:

In the insert operation, you can insert a value at any index. So you have to enter the value and the index number. Then hit "*Next*" button.


Insert Operation
Enter the element you want to insert:
776
Enter the position where you want to insert the element:
4
Next
Back
Go to Array Home

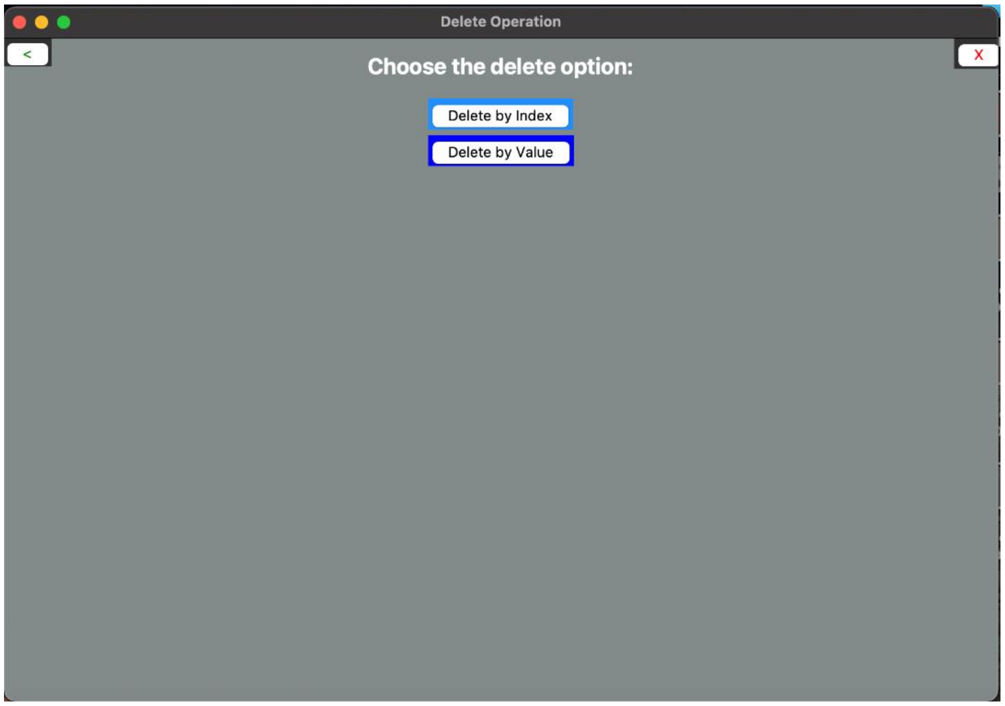
After inserting 776 at position 4:

Updated array: [3, 41, 674, 16, 776, 79]

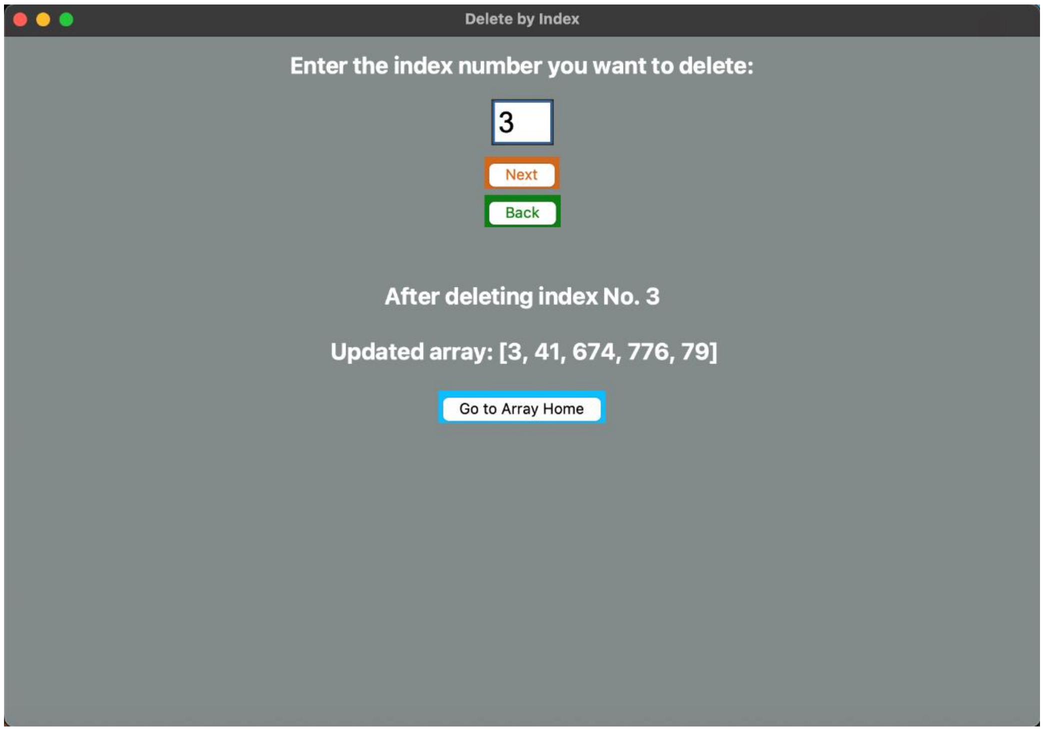
.

Deletion:

In the deletion operation, you can delete elements by indexing and by value.



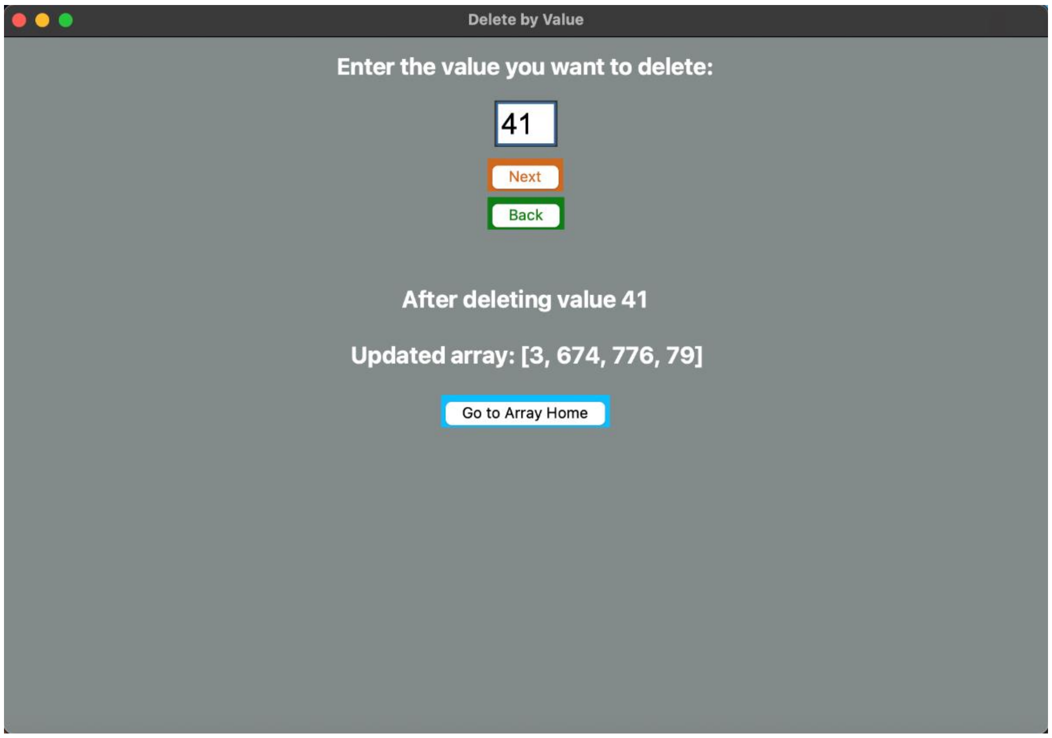
Delete by Index: Enter the index number you want to delete and hit "*Next*" button.



After deleting index No. 3

Updated array: [3, 41, 674, 776, 79]

Delete by Value: Enter the value you want to delete and hit "*Next*" button.

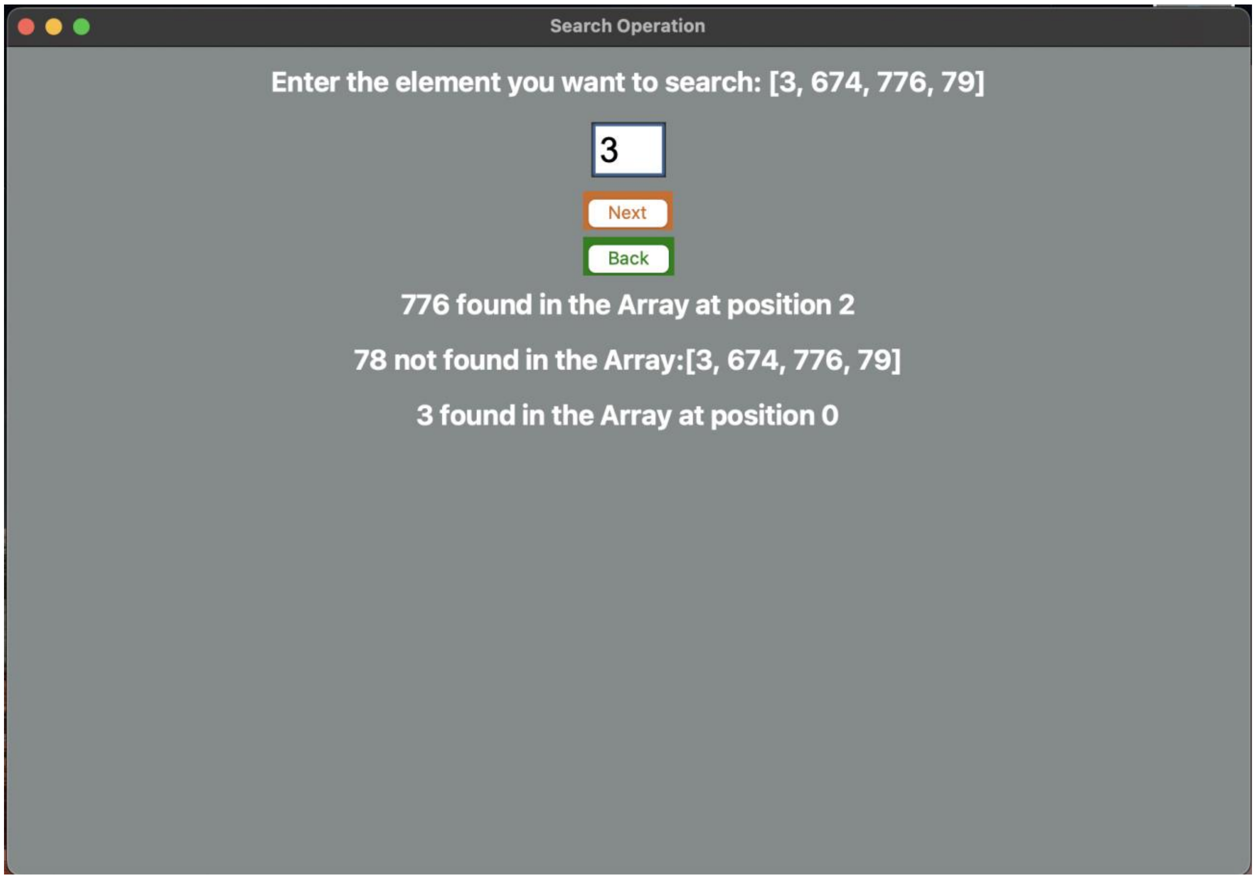


After deleting value 41

Updated array: [3, 674, 776, 79]

Search Element:

Enter the element you want to search in the array and hit "*Next*" button.



Enter the element you want to search: [3, 674, 776, 79]

3

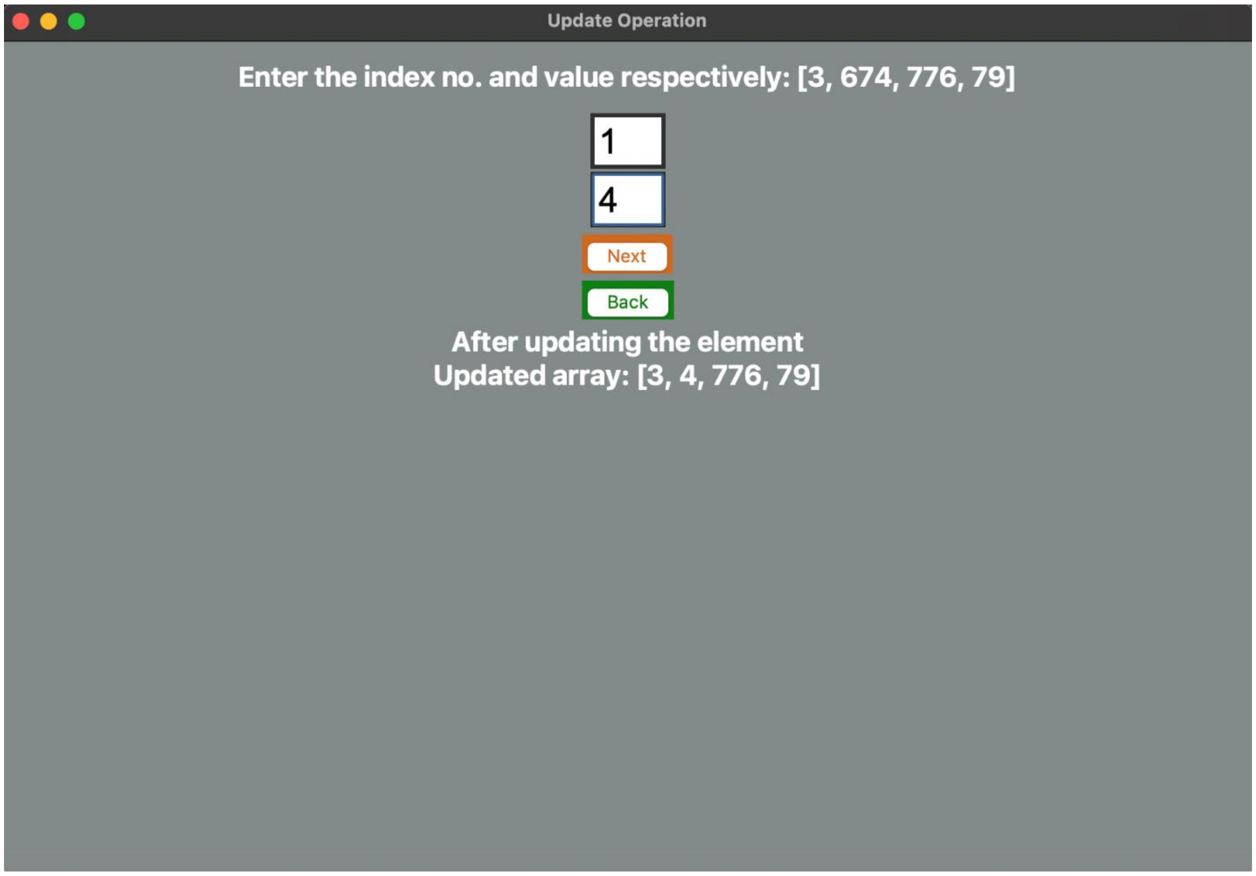
776 found in the Array at position 2

78 not found in the Array:[3, 674, 776, 79]

3 found in the Array at position 0

Update Element:

In the first entry box input the index of the old value that you want to update and in the second entry box enter the updated value and then hit "*next*" button.



Enter the index no. and value respectively: [3, 674, 776, 79]

1

4

After updating the element
Updated array: [3, 4, 776, 79]

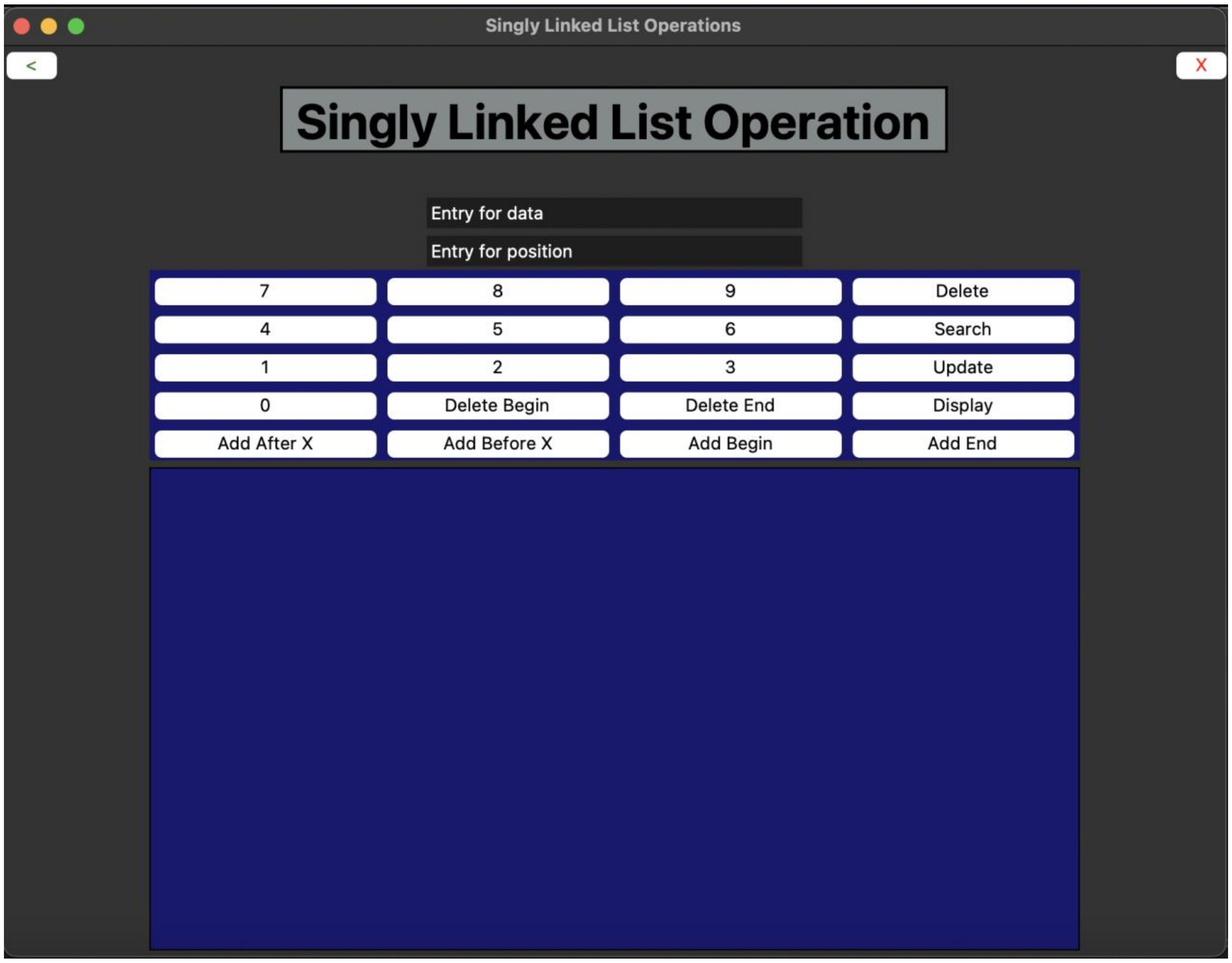
And, the array operation is finished here.

Linked list operation started on the next page…………….

Linked list:

By clicking the "*Linked list*" button, you go to the linked list page.

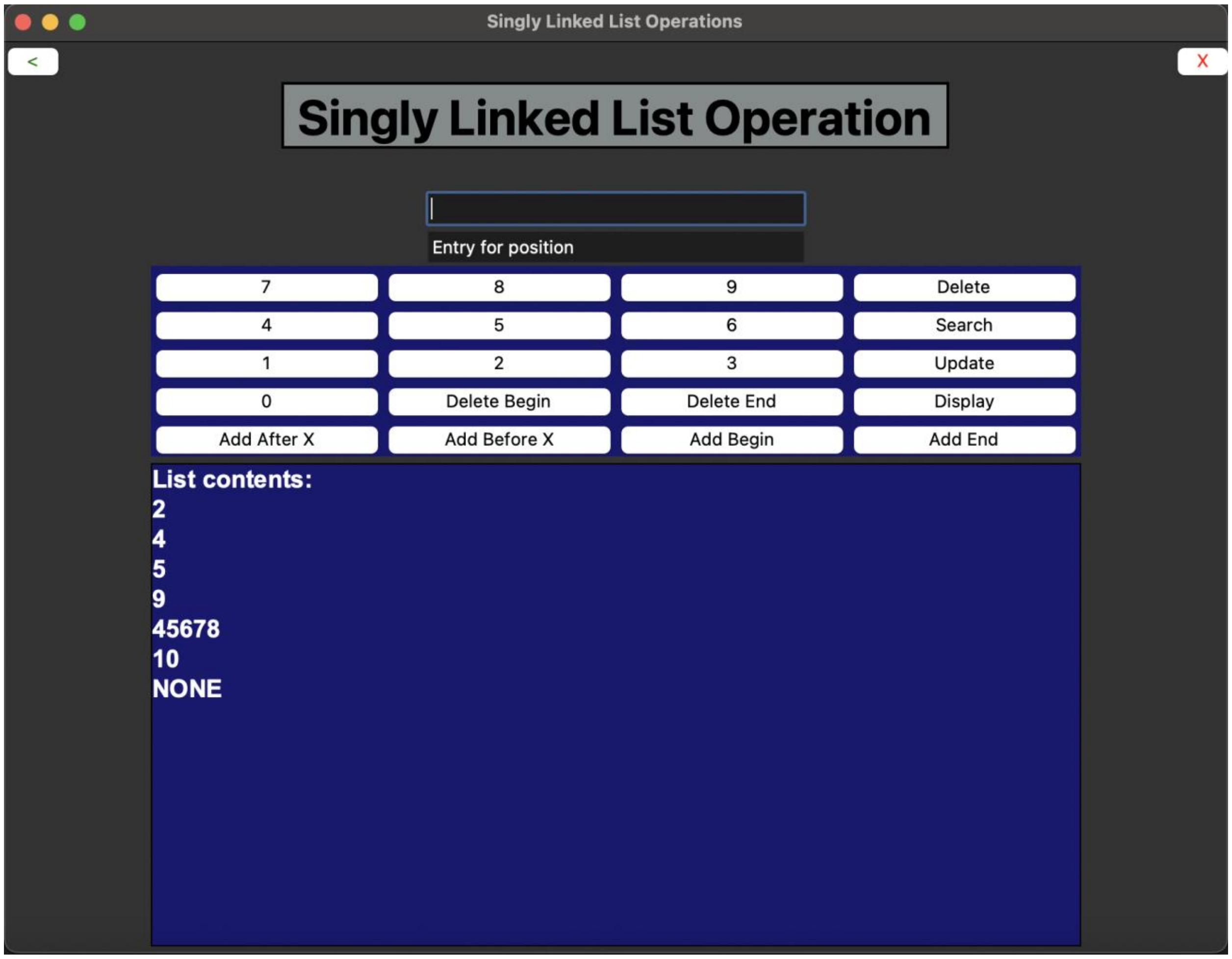
At first, it is an empty linked list.



You can do any operation in the linked list by operating the entry box and these buttons.

Insertion & Deletion:

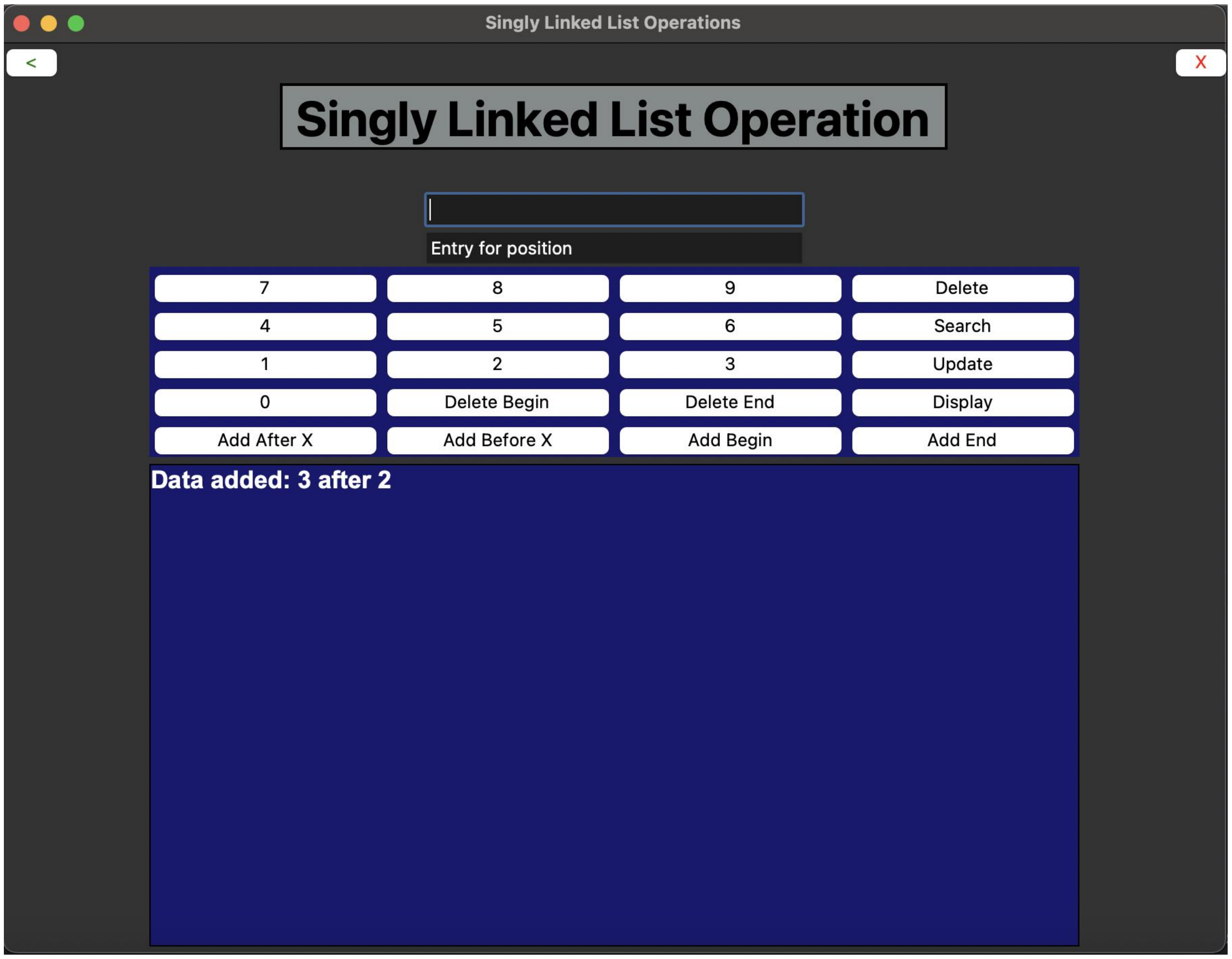
By performing some insertion (at begin, at end) and deletion(at begin, at end) operations the updated linked list picture is given below.



```
List contents:
2
4
5
9
45678
10
NONE
```

Insert after x:

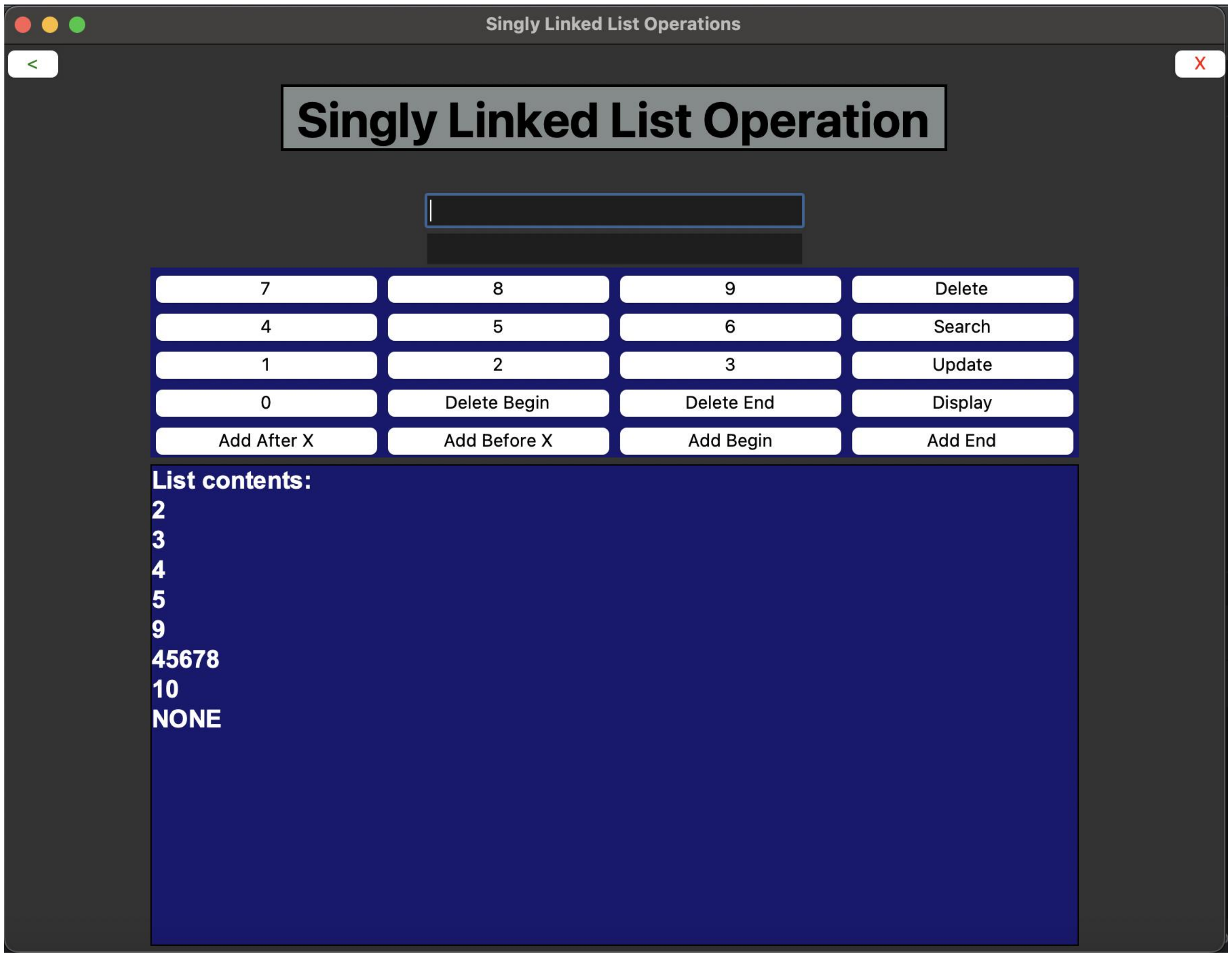
To insert an element after x element, enter the element in the first entry box and enter the x element in the second entry box as shown in the given picture on the next page “3” added after “2”.



Data added: 3 after 2

To check the updated list after adding element after x element, see the next picture.

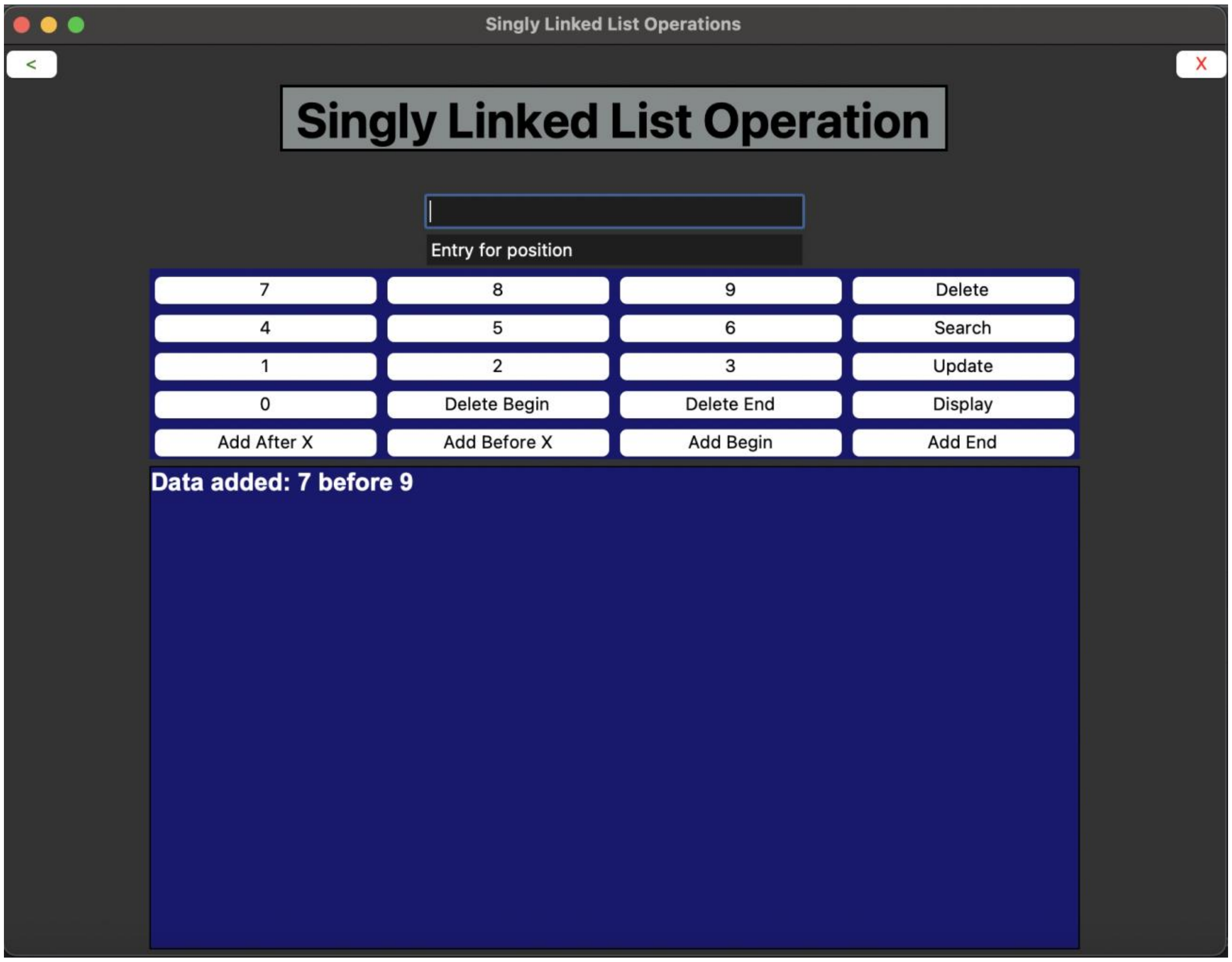
See the linked list that “3” actually added after “2”



```
List contents:
2
3
4
5
9
45678
10
NONE
```

Insert before x:

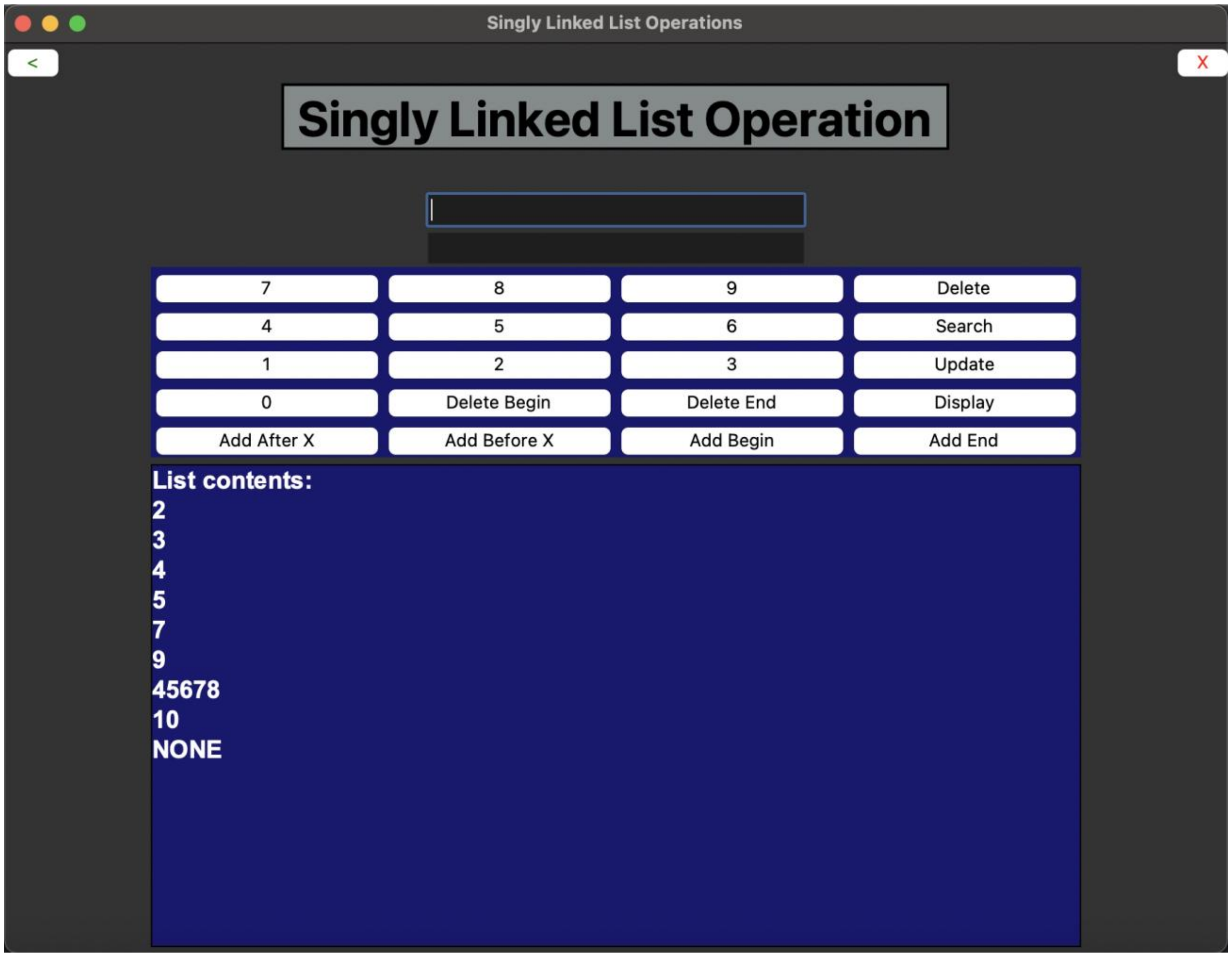
To insert an element before x element, enter the element in the first entry box and enter the x element in the second entry box as shown in the given picture on the next page “7” added before “9”.



```
Data added: 7 before 9
```

To check the updated list after adding the element before x element, see the next picture.

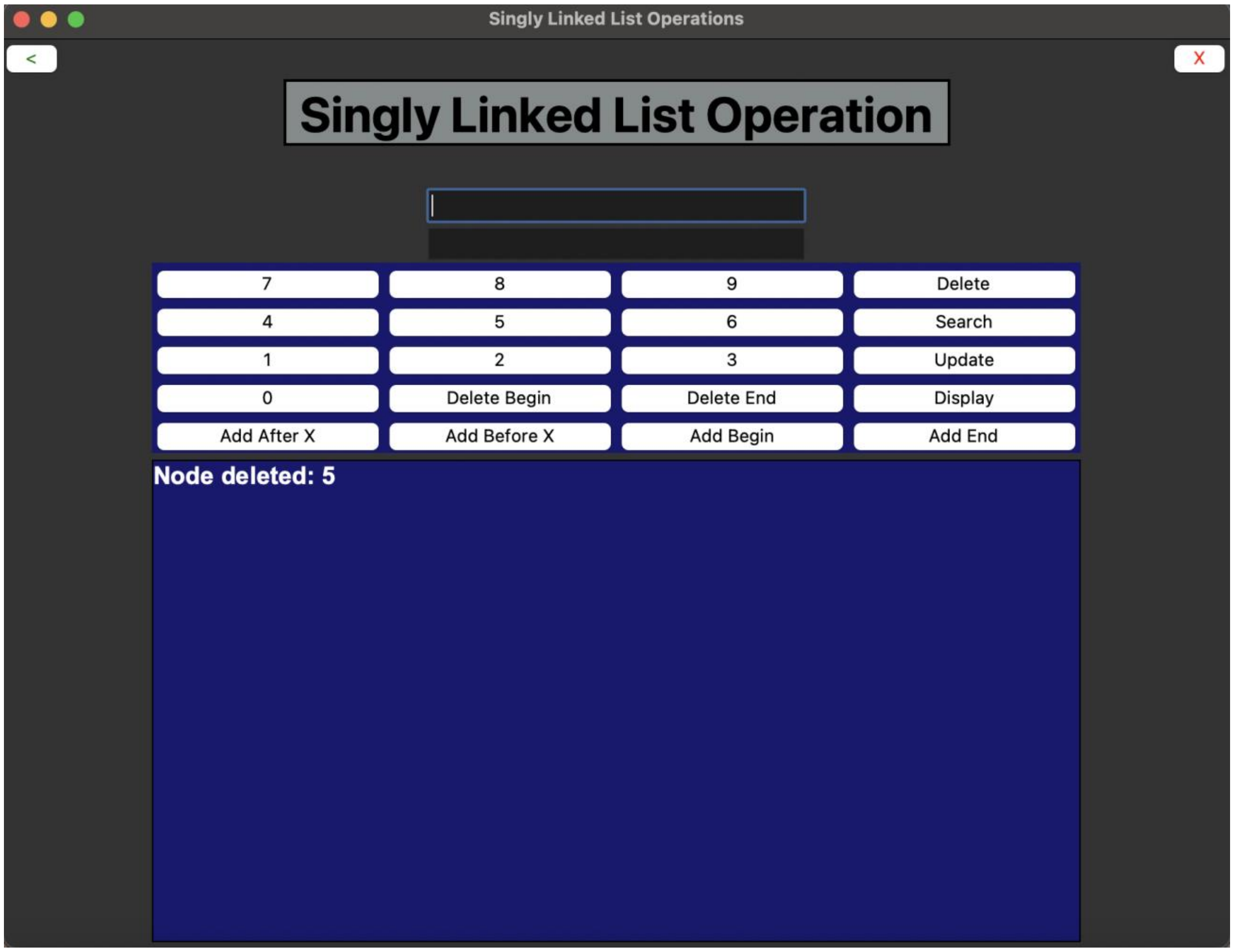
See the linked list that “7” actually added before “9”



```
List contents:
2
3
4
5
7
9
45678
10
NONE
```

Delete:

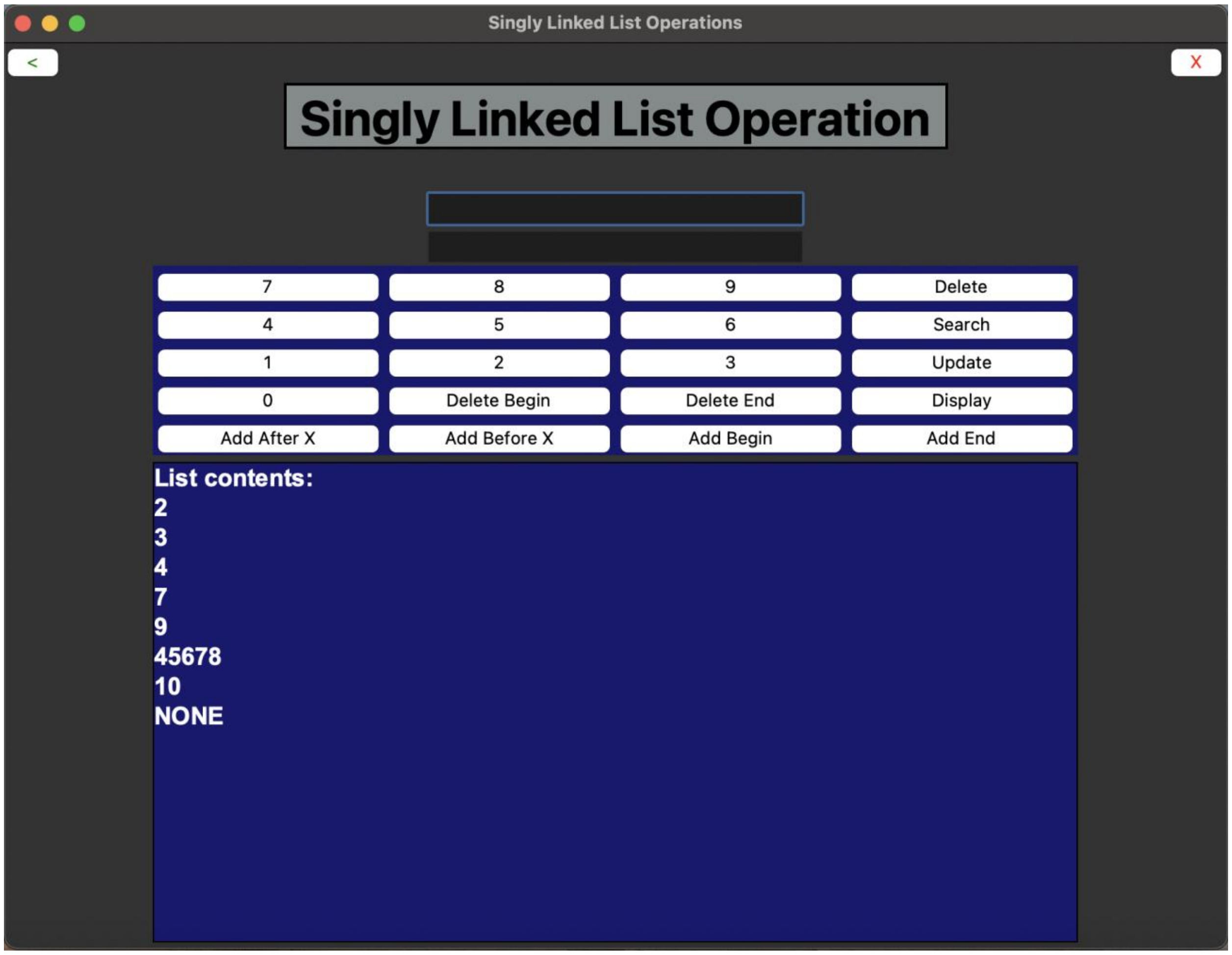
Enter the element in the first entry box that you want to delete and then hit “*Delete*” button. Assume that you enter “5” and then hit “*Delete*” you can see the result as the next picture.



```
Node deleted: 5
```

To check the element actually deleted or not hit “*Display*” button.

See that 5 is no longer in the linked list. “5” deleted



```
List contents:
2
3
4
7
9
45678
10
NONE
```

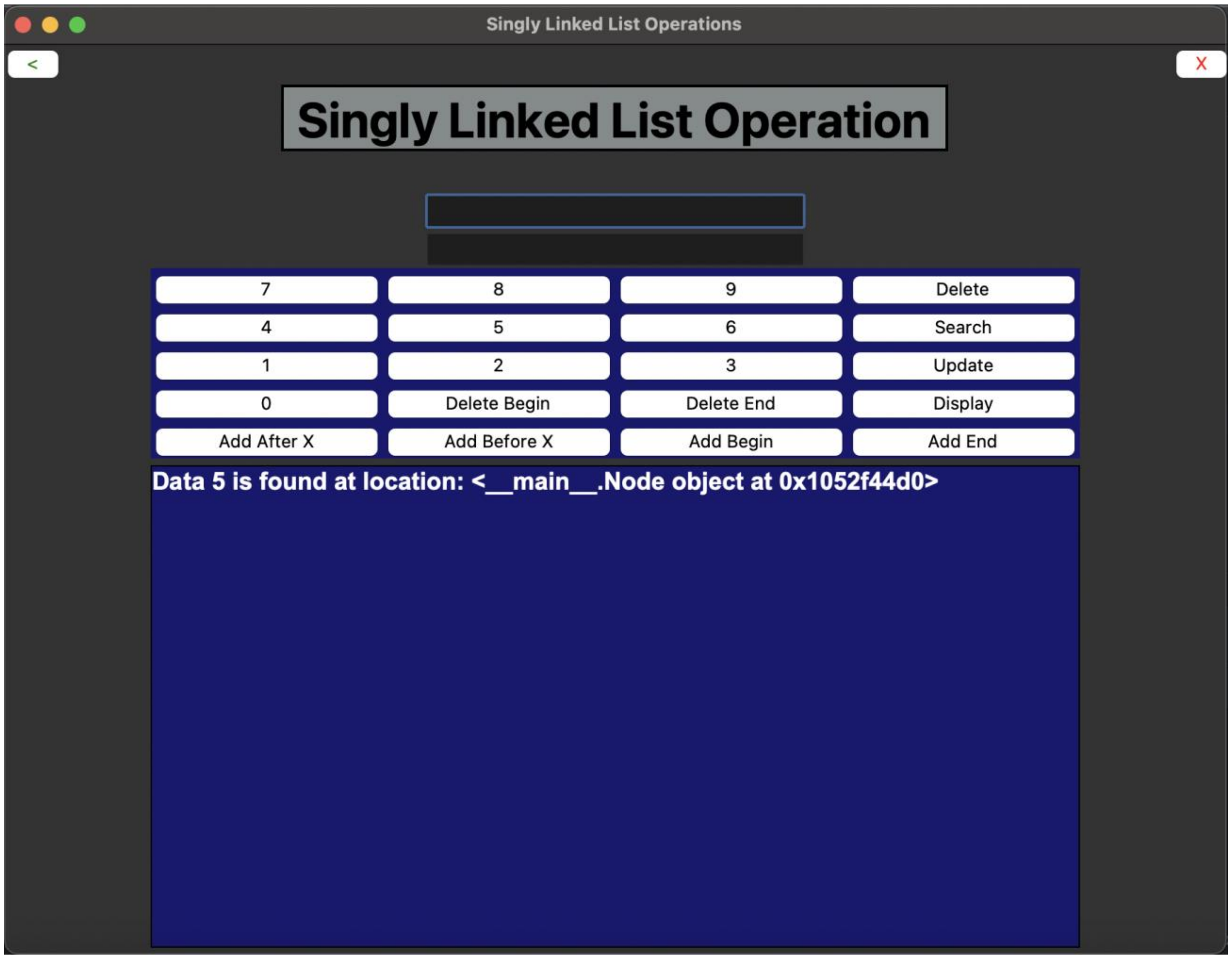
You can also perform **search** and **update** operations in the linked list.

Search element:

Search an element in the linked list that element is present in the linked list or not.

If the element is present in the linked list then where is it located in the memory address?

Enter the element you want to **search** in the linked list then hit "Search" button.

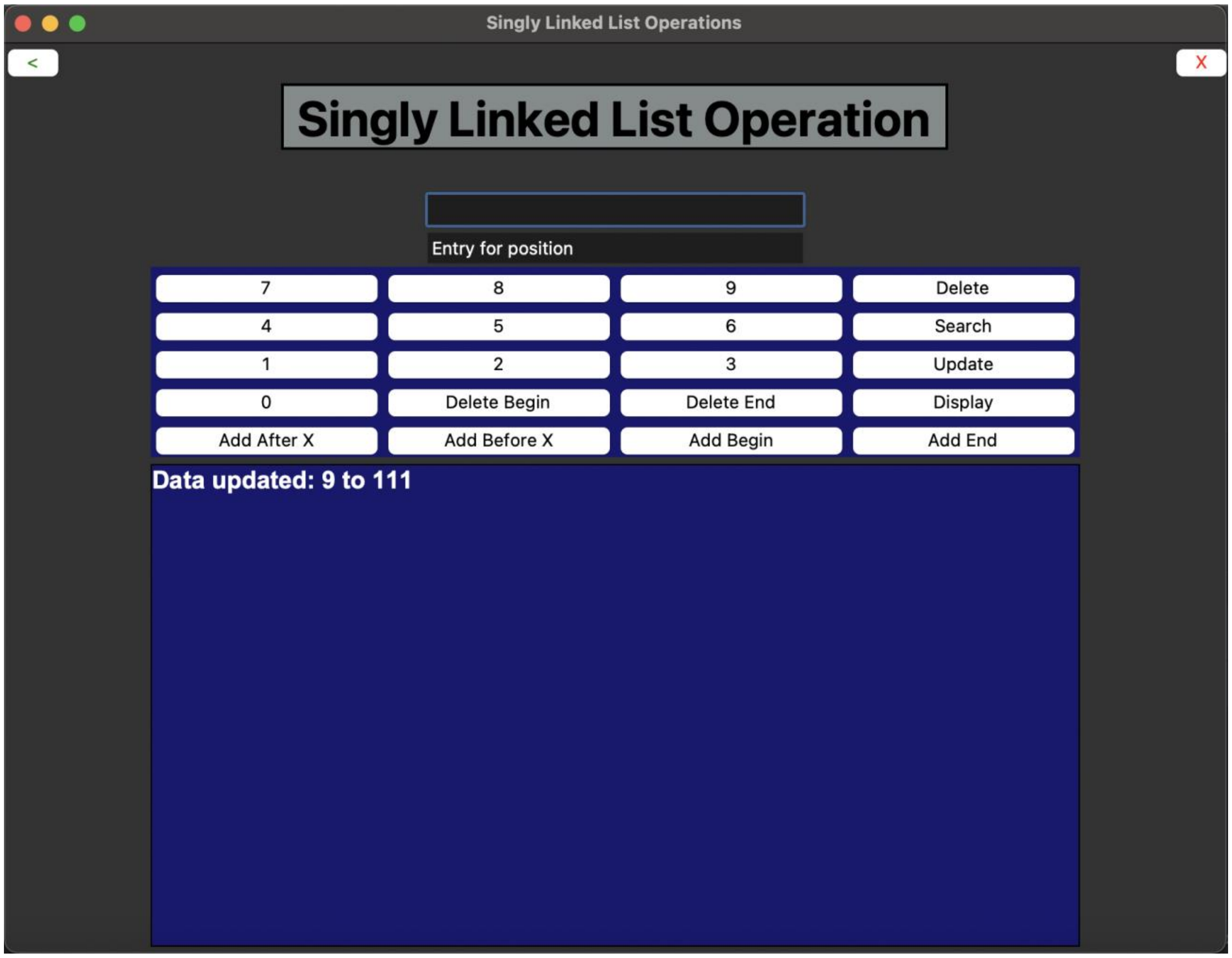


```
Data 5 is found at location: <__main__.Node object at 0x1052f44d0>
```

Assume that you searched "5" and "5" is present in the linked list.

Update element:

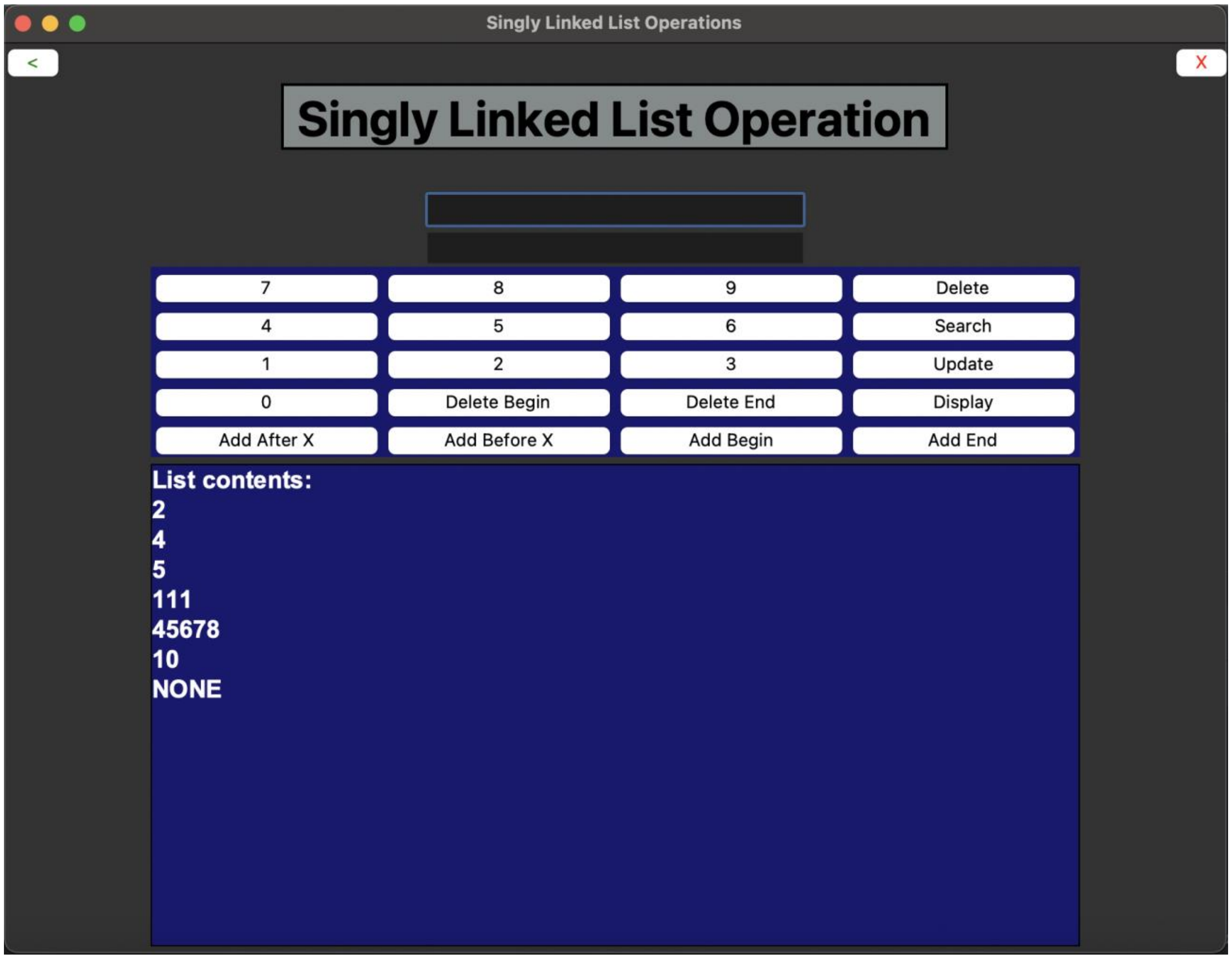
To update an element in the linked list, you have to enter the update element at the first entry box, and what the element you want to update enter the element in the second entry box.



Data updated: 9 to 111

Here “9” updated into “111”

To see the updated linked list you have to hit “Display” button.



```
List contents:
2
4
5
111
45678
10
NONE
```

You can see the element "9" updated into "111"