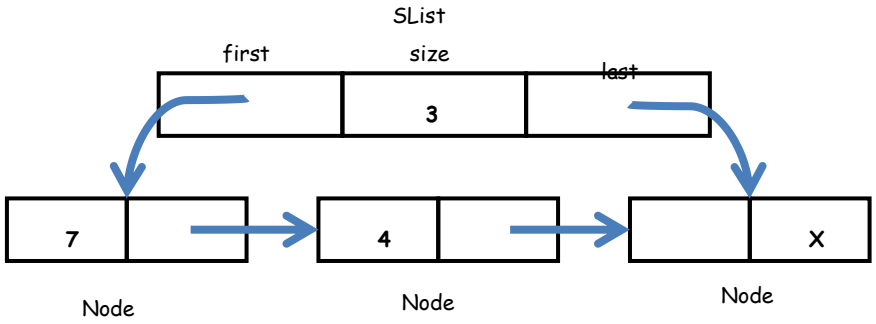
Lab 2: Singly Linked List

Objectives

- To understand the concept of a singly linked list data structure.
- To understand the basic operations related to a singly linked list.
- To write simple applications that use a singly linked list data structure.

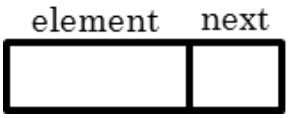
Reviews

- **A singly linked list** is a list of data that are linked or chained in a linear fashion, ie, each element in it except the last one will be followed by another list. Each datum is kept in a node and each node has an information of which node is next to it.
- **A Singly Linked List (SList)** composes of nodes that keep data and links that tie up two a node and its successor together.
- **A singly linked list model**



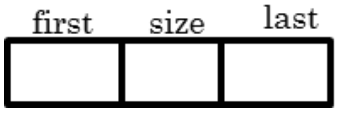
Node UML

Node<T>
element: T next: Node<T>
Node(T element)



SList UML

SList<T>
size: int first: Node<T> last: Node<T>
SList() addFirst(element: T): void addLast(element: T): void addAtIndex(index: int, element: T): void removeFirst(): T removeLast(): T removeAtIndex(index: int): T isEmpty(): boolean getSize(): int search(T item): int printHorizontal(): void



Basic add operations

- The addFirst(T element) and addLast(T element) methods add new element to the head of the list and the tail of the list, respectively. Fig 1. Illustrates the addLast operation.
- The addAtIndex(int index, T element) method adds a new element at the specified index in the list (**the index of the first element is 0**)

Basic remove operations

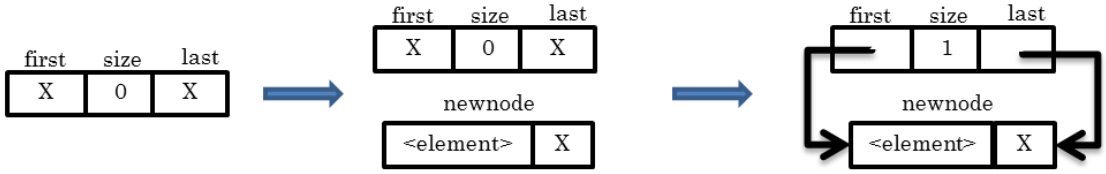
- The removeFirst() and removeLast() methods remove the first node and last node of the list, respectively. These methods return the element that is contained in the removed node.
- The removeAtIndex(int index) method removes the element at the specified position in the list and returns the element that was removed from the list.

Basic search operation

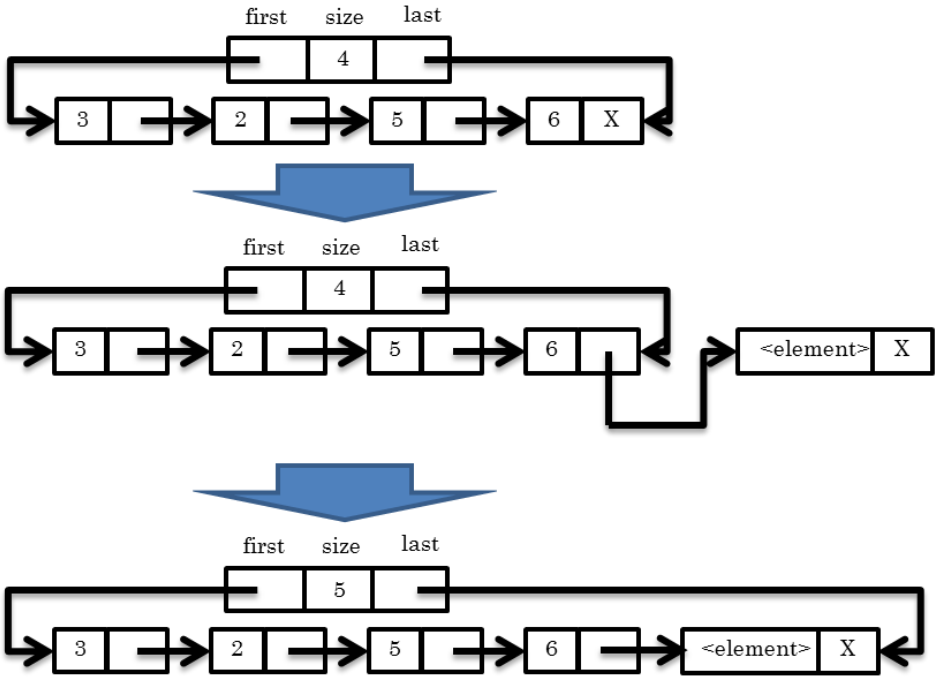
- The search (int item) returns the index of the first Node that contains the item. If the item can't be found in the list, the method returns -1. Test the method in main().

AddLast Illustration

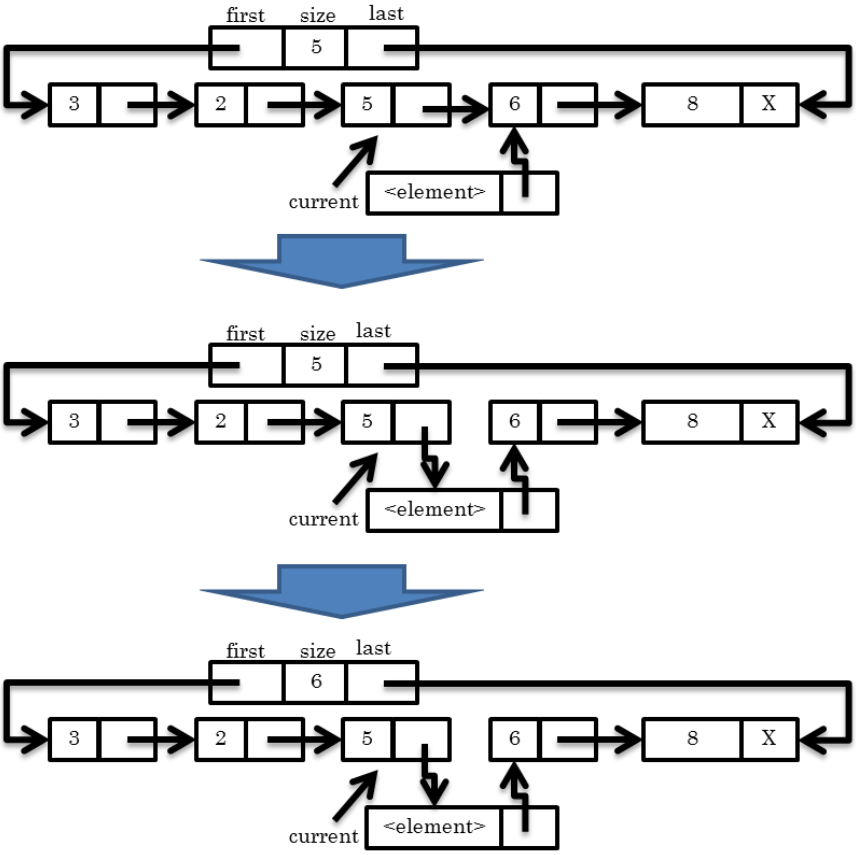
Case 1: No node in the list



Case 2: There are node(s) in the list



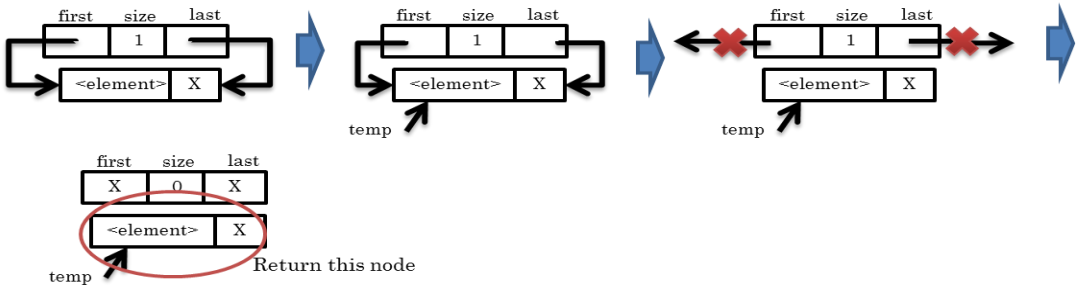
AddAtIndex Illustration



RemoveLast Illustration

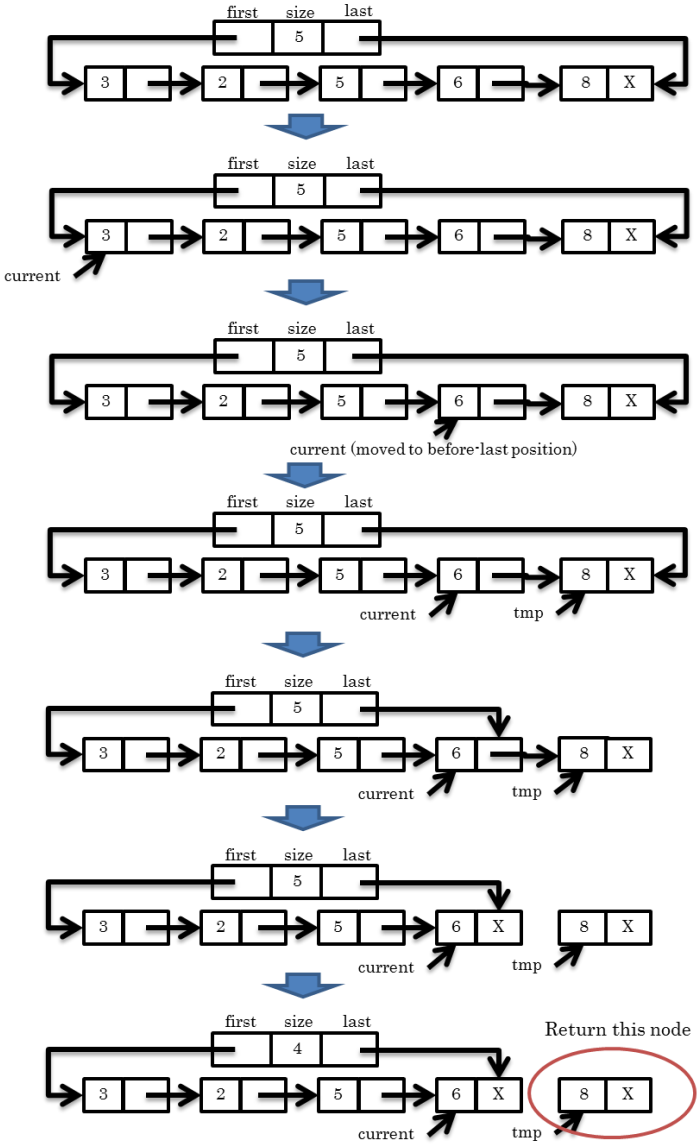
Case 1: No element in the list -> return null

Case 2: 1 element in the list



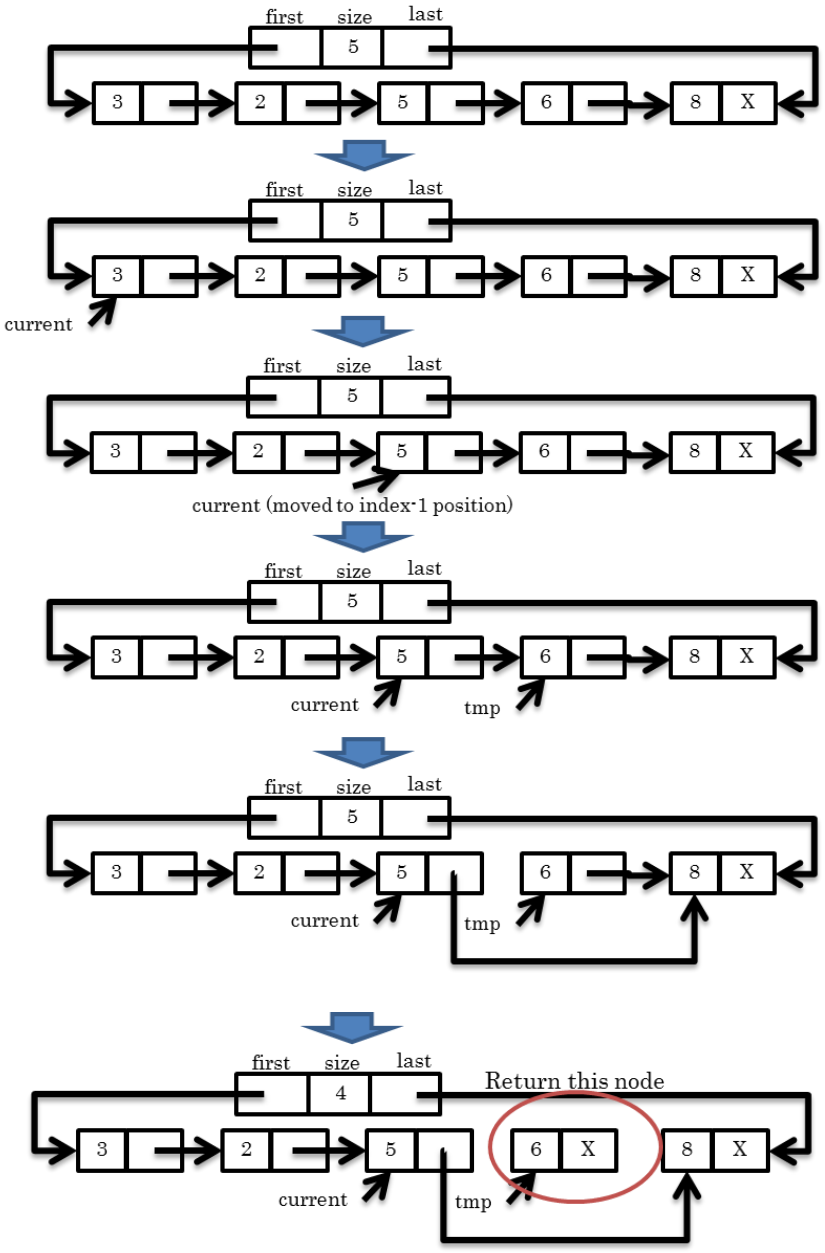
Case3: More than 1 element in the list

Let assume that list's size = 5



RemoveAtIndex Illustration

Let assume that list's size = 5 and index = 3



In this section, you will implement methods that are used with singly linked list data structure and test them in main() method.

Exercise 1 The method addFirst is already given in the template. In this exercise, implement addLast(T element). Test the methods in your main() method. Use the given template source code to complete this exercise. You may use the following figures for the implementation.

Exercise 2 Implement addAtIndex(int index, T element). Test the method in your main() method. Use the given template source code to complete this exercise.

Exercise 3 The method removeFirst is already given in the template. In this exercise, implement removeLast(). Test the methods in your main() method. Use the given template source code to complete this exercise.

Exercise 4 Implement removeAtIndex(int index). Test the method in main(). Use the given template source code to complete this exercise.

Exercise 5 Implement the method int search(T item). The method returns the index of the first Node that contains the item. If the item can't be found in the list, the method returns -1. Test the method in main().