# Assignment 4 --- Lists & Recursion

*This assignment has two parts; the first part covers lists where the goal is to construct and visualize a new Singly Linked List implementation using Bridges SLelements instead of Linked Node objects. The second part covers recursion in which you will be asked to design four recursive functions.*

## Part I

### Overview

The primary goals of this program are as follows:

(1) Create a new implementation of the List class using Bridges SLelement objects instead of the Node class we created in class.

(2) Create a Tester class to demonstrate the Singly Linked List data structure. Assign visual properties to the elements of the List.

The **Singly Linked List implementation** will be almost exactly the same as the Linked List implementation we worked on in class. Singly Linked Elements from the Bridges package support the same operations as the Node class, although the names for the various operations might differ slightly.

You essentially need to go through the Linked List implementation, find every occurrence of a Node object and its operations, and replace it with the equivalent SLelement objects and operations. The logic of the data structure should not change (provided your original LinkedList class worked properly).

The **Tester** should create an instance of the Singly Linked List class. The list will store a sequence of integers. The list should be visualized a number of times as elements are added to it, and each set of new elements should feature different visual properties. Visualizing the data structure and adding visual properties to list elements can be accomplished easily with the Bridges API. See the Tasks section for specific details and instructions.

### Tasks

**Follow** the instructions on this website to add the Bridges library to BlueJ:
http://bridgesuncc.github.io/bridges_setup_java_bluej.html

**Create** a project named Training in BlueJ and re-create this example:
http://bridgesuncc.github.io/Hello_World_Tutorials/SLL.html

**Documentation** for Bridges classes can be found at the following link:
http://bridgesuncc.github.io/doc/java-api/current/html/index.html

**Singly Linked List Assignment–**

--- Create a ourLinkedList interface with the following functions:

- add (list, element): void – this method appends the specified element to the list
- contains (element): boolean – this method returns true if the list contains the specified element
- peek (): T
- remove (index): T – this method removes the element at the specified index and returns that element
- findRightPlaceAndInsert(element): Boolean – this method finds the right place for the element based on chronological order then add the object to the list at that place. The method returns true when the object is added to the list successfully.
- findAndHighlight (element): true -- this method returns true if the element is in the list. The method change the size and color of the node holding that element.

--- Create SList class that implements the LinkedList interface (you will need to implement the node class inside the SList class)

--- Import *bridges.base.SLelement library* to allow you to use SLelement objects in this class

--- Replace every instance and operation of the Node class with an SLelement object and its equivalent methods

--- Don't forget to parameterize the SLelement objects with the generic type argument. You can examine the documentation to determine which methods to use to replace the Node class' methods

--- Create an updateLabel method that sets the *label* equal to the new element's value

--- Create methods to change the current element's color, size, shape, and opacity. These are all supported with Bridges methods accessible through SLelement's *getVisualizer* method. Since the driver won't have direct access to the list's *current* SLelement, you need to call these respective Bridges methods from the SLList class using arguments passed from the driver.

----

**Tester –**

--- Create a Tester class with a main method

--- Import *bridges.connect.Bridges* to allow you to create a Bridges object

--- Initialize a Bridges object with the assignment number, your username, and your API key. (See the Bridges template on Moodle for details)

--- Create an instance of your SList class and parameterize it to hold Integers

--- Add 100 new elements to your SList. The elements should simply contain the index of whichever loop construct you use, and should be inserted in increasing order (from 0 to 99)

--- Every successive set of 25 elements should have different visual properties from the other 75. You are free to use whichever attributes you like:

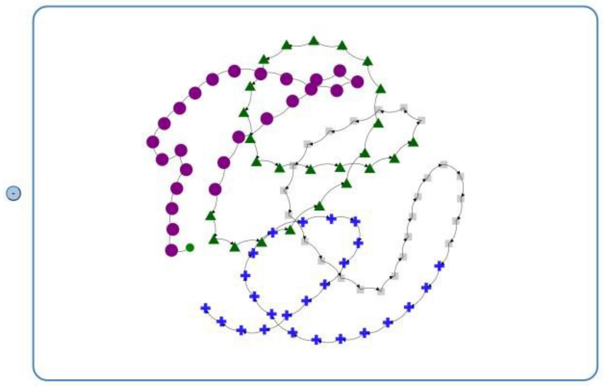
- o color can be set to any css or RGB color
- o opacity can be set between 0 and 1
- o size can be set between 1 and 50
- o shape can be set from among the following options: [square, diamond, triangle-•-down, cross, triangle-•-up, circle]

---

--- After every successive set of 25 elements are added to the list, call Bridges' setDataStructure method passing in the head of your list and then call Bridges' visualize method

**Deliverables –**

The final Bridges assignment your program should generate will be something like this:



**Scoring Rubric**

**Tester:** 80 points

--- Up to 10 points for appropriate documentation and comments
--- Up to 10 points for correctly adding 100 elements to the list in increasing order
--- Up to 20 points for setting the labels for all the added elements
--- Up to 20 points for setting new distinct visual properties for each new set of 25 elements
--- Up to 20 points for visualizing the data structure after each set of 25 elements are added (you will generate a total of 4 visualizations with 25, 50, 75, and 100 elements, respectively)

**Singly Linked List:** 50 points

--- Up to 10 points for appropriate documentation and comments
--- Up to 20 points for correctly implementing the Singly Linked List class with SLelement Nodes
--- Up to 20 points for correctly adding methods to set color, opacity, shape, and size for the current element

**Total points available: 130**