**PROGRAMMING PATTERNS**

420-301-VA

#### **LAB-2**

### Instructions

* Labs’ answers must be demoed to the teacher in class.
* Labs are individual works.
* You are supposed to work on this LAB during the session hours otherwise you risk falling behind.
* First, try to solve each problem without assistance except asking clarifying questions, preferably from the teacher.
* Do not look at solutions or get assistance from colleagues, online resources, or AI before spending considerable time trying to do the work from scratch yourself, some problems may require hours to solve, and you may need to look away work on another assignment and get back to the problem with a fresh eye that is normal and differs from individual to individual.
* The lab work is based on the theory sessions and a direct implementation of the concepts. Make sure you understand the concepts and can do the examples, or follow the logic, discussed in class and identify what the lab is asking you to do relative to the discussed concepts.
* Feel free to ask more questions to the teacher or look at different examples that are not the solutions to the lab from any other resource.
* Note that the due date is different for each section.
* Note that if you could not work on a lab by the deadline, you should still work on it, submit it and discuss the case with the teacher.

## Data Structure Definition

1. Using the definitions writing technique discussed in class write the definition of a node-based data structure (research information if needed):

|  |
| --- |
| **What are the characteristics of a node-based data structure?**   * **Nature**: it is a structure that organizes data in an individual element called a node where each node contains a data type and is connected to other nodes. * **Purpose**: it allows efficient data searching, manipulation, and modification. * **Composition/elements**: it contains a data, a next element, and a previous element. In a list, a node can have one pointer that points to a previous node and a second pointer that points to the next node. In a tree, we can have multiple pointers pointing to the next node. * **Usage**: it is used as a foundation concept for data structure implementation such as linkedlist, stack, queue, etc. * **Properties**: The data storage can dynamically grow and shrink. It is also very flexible as it can be used to represent many different relationships. * **What it is not**: it is not an array * **Examples (give a specific example of a group of data that could be stored in the array)**:  binary tree |

|  |
| --- |
| **Write a full comprehensive statement defining a node-based data structure putting all the characteristics listed above together in a comprehensive statement(s):**  **A node-based structure is a structure that organizes data into individual element called a node. Each node contains a data type, and it is connected to other nodes. Node-based allows efficient data traverse, searching, manipulation as well as modification. A node has a data, and a pointer that points to the next node. In a list, a node can have one pointer that points to a previous node and a second pointer that points to the next node. In a tree, we can have multiple pointers pointing to the next node. A node-based structure is used as a foundation concept that is applied to data structure such as linkedlist, stack, queues, and so on. Data storage of a node-based structure can dynamically grow and shrink. It is also versatile as it can be used to represent many different relationships, not just linear relationships. A node-based structure does not include array as it doesn’t use nodes. For example, a binary tree.** |

## Node Based Data Structure Operations

Considering the SingleLinkedList class that we have built with its add(E e) method implement the following additional methods:

Note you can use the attached code as a starting point if you haven’t implemented the add method yet.

|  |  |
| --- | --- |
| Object[] toArray() | Returns an array containing all of the elements in this list in proper sequence (from first to last element).  **Implement the code in the main class that would test this method by printing all the elements of the list.**  **Use this code to test the implementation of the remaining methods below.** |
| void addLast(E e) | Appends the specified element to the end of this list. |
| void add(int index, E element) | Inserts the specified element at the specified position in this list. |
| E getFirst() | Returns the first element in this list. |
| E getLast() | Returns the last element in this list. |
| E get(int index) | Returns the element at the specified position in this list. |
| int indexOf(Object o) | Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element. |
| E remove() | Retrieves and removes the head (first element) of this list. |
| E remove(int index) | Removes the element at the specified position in this list. |
| boolean remove(Object o) | Removes the first occurrence of the specified element from this list, if it is present. |
| int size() | Returns the number of elements in this list. |
| boolean contains(Object o) | Returns true if this list contains the specified element. |