# Stack and Queue

## Part 1 Description

Create two new files named Stack.java and Queue.java which will contain classes for a stack and a queue using a LinkedList to store their data. The LinkedList class is given already and should be fully functional. Each class will need to take a generic data type, T.

### Stack

Stack will need three methods.

1. **push** method will take a T parameter and store it in the LinkedList.
2. **pop** method will remove a T item from the LinkedList in correct stack order and return it.
3. **getList** method will return the LinkedList being used by the stack (this is required for testing)

### Queue

Queue will need three methods.

1. **add** method will take a T parameter and store it in the LinkedList.
2. **remove** method will remove a T item from the LinkedList in correct queue order and return it.
3. **getList** method will return the LinkedList being used by the queue (this is required for testing)

You may feel free to add other helper methods to your classes, but they are not required. **Do not change LinkedList.java and do not use any built in Java data types.**

## Part 1 Starter Code

Download the start code **LinkedList.java** from the course public directory ([public/final](https://cs.unh.edu/~cs416/public/final))

# NodeList Find All

## Part 2 Description

Write a method findAll which takes a String parameter and returns an ArrayList of Integer objects. Note that this linked list's nodes each contain a String key and an int data field. Your findAll method should return an ArrayList containing the data from any node whose key matches the key passed in as a parameter to the method.

## Part 2 Starter Code

Download the start code **NodeList.java** from the course public directory ([public/final](https://cs.unh.edu/~cs416/public/final))

# 

# BST findAll

## Part 3 Description

Write a **findAll** method for the given BST class, which will take an array of int keys and returns an array of String values. Note that this BST's nodes each contain both a key and a value, and the BST is ordered by keys. The method should attempt to find each key in the parameter array and set the value in the return array at that same index to the result. The return array should have null at the index for any key which isn't found in the tree.

For example, in the tree shown below, findall(new int[]{1, 4, 3, 5}) should return the array of strings {null, "B", "H", "G"}.

L (2, D)

R (3, H)

L (4, B)

R (5, G)

(7, A)

R (8, C)

L (9, F)

R (12, E)

You may feel free to add other helper methods to your classes, but they are not required. **Do not change any of the other methods of the BST class.**

## Part 3 Starter Code

Download the start code **BST.java** from the course public directory ([public/final](https://cs.unh.edu/~cs416/public/final))