Suppose we have the following definition of a Node:

```
class Node<T> {
  public T data;
  public Node<T> next;

  public Node<T>(T data, Node<T> next) {
    this.data = data
    this.node = node
  }
}
```

Diagramming Question

Suppose that I have the following code. Diagram the objects allocated and the links between them.

```
Node<String> a = new Node<String>(“a”, null);
Node<String> b = new Node<String>(“b”, a);
Node<String> c = new Node<String>(“c”, b);
Node<String> d = new Node<String>(“d”, c);
Node<String> e = new Node<String>(“e”, d);
Node<String> f = new Node<String>(“f”, d);  // Note the difference here!
```

Implementation Question

Write a function called average which gets the average of a singly linked lists of doubles (or more specifically Double, which is a pointer to a double and acts like a double for all purposes here). We are given the head node. It will likely return a divide by zero error if head is null:

```
public double average(Node<Double> head) {




}
```

What is the big-Oh running time of your function?

Implementation Question

Write a function called average which gets the average of a singly linked lists of doubles (or more specifically Double, which is a pointer to a double and acts like a double for all purposes here). We are given the head node. It will likely return a divide by zero error if head is null:

```
public double average(Node<Double> head) {




}
```

What is the big-Oh running time of your function?

Diagramming Question

Suppose that I have an array of arrays (remember that's an array of pointers, all pointing to other arrays)

int[][] arr = { {1, 2, 3}, {}, null, {10, 11, 12} };

Diagram how this is stored in memory.

Running Time

What is the big-Oh running time of the following function in terms of n?

```
int countingOnAStack(int n) {
  Stack<Integer> stack = new ArrayStack(1000);
  stack.push(n);
  return countingOnAStack(new ArrayStack(1000);
}

int countingOnAStack(Stack<Integer> stack) {
  while (!stack.isEmpty())
    int item = stack.pop();

    if (item >= 0) {
      stack.push(item - 1);
      stack.push(item - 1)
    }
  }
}
```

Running Time

What is the running time of the following code in terms of n?

```
// arr is an array of length n
int currentMax = 0;
for(i = 0; i < n; i++) {
  if (arr[i] > currentMax) {
    currentMax = arr[i];
  }

  // Increase the rest of the array
  for(j = i + 1; j < n; j++) {
    arr[j] += 1
  }
}
```

Running Time + Coding

What is the worst case running time of findAll in terms of n, which is the length of an array?

```
//  int[] arr is an array of length n

int findInArray(int element) {
  for(i = 0; i < n; i++) {
    if (element == i) return i;
  }
  return i;
}

int[] findAll() {
  int[] answer = new int[n]; // Allocate an array of length n

  for(int i = 0; i < n; i++) {
   answer[i] = findInArray(i)
  }

  return answer?
}
```

Could you write a better version of findAll that accomplishes the same thing with a better big-Oh running time?

Coding

You are given a linked list. Construct another Linked List that is your input in reverse order.

What is the big-Oh running time of your algorithm?

Stacks (Basic)

What gets printed out by the following code?

```
Stack<Integer> stack = new ListStack<Integer>()
stack.push(2)
stack.push(8)
int x = stack.pop()
stack.push(10)
stack.push(x)
stack.push(x)
stack.push(12)
x = stack.pop();
stack.push(x + 1)

while(!stack.isEmpty()) {
  System.out.println(stack.pop())
}
```

Queue (Basic)

You have a queue of Integers stored in a circular array of length 5. Draw the values of the Queue after the following operation. Shade in the space in the array that is unused.

Queue<Integer> q = new ArrrayQueue(5);

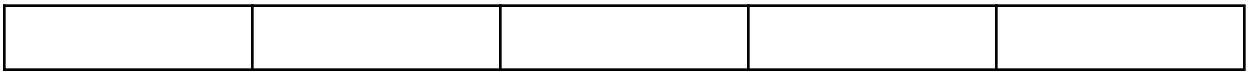
q.enqueue(4)
q.enqueue(3)
q.enqueue(2)

q.dequeue()
q.enqueue(8)
q.enqueue(10)

int x = q.dequeue()

q.dequeue()
q.enqueue(x)
q.enqueue(x * 2)

(also draw the starting and ending elements of the queue)



Coding (Stack)

Given a stack with at least two elements, implement addAndRestack, which pops 2 elements, adds them together, and pushes the sum back onto the stack.

Write a sumUntil(int stop) function that calls sum on the stack until a stopping value is found. In that case, leave the stopping value on the stack, and push the sum on top of it.

Code (Stack)

You are given a headed Singly-linked list, and nodes a and b.

Describe a function called exchange(Node<> a, Node<> b) which exchanges the place of the nodes a.next and b.next. You algorithm may only modify links, and may not allocate new nodes (you must work with the nodes you have, only changing the next values)

What is the Big-Oh running time of your algorithm?

Array Scheme

You have an array of arrays.

int[][] arr.

The array is of length n, and arr[i] is always of length 2^i

Describe a scheme to store this entire data structure in a single array, and how you would access arr[i][j] in this single array.