Recursion – pt 2

Ch. 12

```
public class PrintList{
   public static void main(String[] args) {
      Node head = null;
      for (int i = 5; i < 10; i++){
         head = new Node(i, head);
      }
      traverse(head);
   }
   static void traverse(Node list) {
      if (list != null) {
         // what happens if we switch the order?
         traverse(list.next);
         System.out.println(list.data);
      }
   }
}
```

```
// Given the linked list below, complete the method sum() that
// calculates the sum of values in the linked list. Feel free to
// use iteration or recursion.

public class RecurTest{
   public static void main(String[] args){
      Node head = null;
      for (int i = 0; i < 5; i++){
         head = new Node(i, head);
      }
      System.out.print(sum(head));
   }
   public static int sum(Node h){
   // Your code goes here




   }
}
```

```
// Binary search using recursion
// Caveat – x must be in sorted order
int binarySearch(int[] x, int start, int end, int n) {
   if (end < start){
      return -1;
   }
   int mid = (start+end) / 2;
   if (x[mid] == n) {
      return mid;
   }
   else {
      if (x[mid] < n) {
         return binarySearch(x, mid+1, end, n);
      }
      else {
         return binarySearch(x, start, mid-1, n);
      }
   }
}
```

Array List

- `add()`
  - `add(item)` – adds item to the end of a list
  - `add(n, item)` – adds item to nth position in list
- `get()`
  - `get(n)` – returns element at index n
- `set()`
  - `set(n, item)` – sets nth element to item
- `indexOf()`
  - `indexOf(item)` – returns index of first occurrence of item
- `remove()`
  - `remove(n)` – removes element at index n
  - `remove(item)` – removes element at first occurrence of item