Recursion (cont)

Midterm
- Next Thursday in class
- Material: everything up to, and including, recursion.
- Closed books. No need to memorize methods of Java classes – whenever required, their interface will be provided.
- You will need to write methods/classes, find errors in code.
- Study lecture slides, code examples, recitation examples, quizzes (and books)

Recursion - examples
- Problem: given a string as input, write it backward
- Base case?
- Recursion

Dictionary lookup
- Suppose you’re looking up a word in the dictionary (paper one, not online!)
- You probably won’t scan linearly thru the pages – inefficient.
- What would be your strategy?

See chapter 3 in walls and mirrors

Binary search

```java
binarySearch(dictionary, word) {
  if (dictionary has one page) // base case
    scan the page for word
  else // recursive case
    open the dictionary to a point near the middle determine which half of the dictionary contains word
    if (word is in first half of the dictionary) { // binarySearch(first half of dictionary, word)
    }
    else { // binarySearch(second half of dictionary, word)
    }
}
```

Binary search
- Write a method `binarySearch` that accepts a sorted array of integers and a target integer and returns the index of an occurrence of that value in the array.
- If the target value is not found, return -1

<table>
<thead>
<tr>
<th>4</th>
<th>2</th>
<th>7</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>22</th>
<th>25</th>
<th>30</th>
<th>36</th>
<th>42</th>
<th>50</th>
<th>56</th>
<th>68</th>
<th>85</th>
<th>92</th>
<th>103</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>2</td>
<td>7</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>22</td>
<td>25</td>
<td>30</td>
<td>36</td>
<td>42</td>
<td>50</td>
<td>56</td>
<td>68</td>
<td>85</td>
<td>92</td>
<td>103</td>
</tr>
</tbody>
</table>

```java
int index = binarySearch(data, 42); // 10
int index2 = binarySearch(data, 66); // -1
```
### Binary search

```java
// Returns the index of an occurrence of the given value in the given array, or -1 if not found.
// Precondition: a is sorted
public static int binarySearch(int[] a, int target) {
    return binarySearch(a, target, 0, a.length - 1);
}

// Recursive helper to implement search.
private static int binarySearch(int[] a, int target, int first, int last) {
    if (first > last) { // not found
        return -1;
    } else {
        int mid = (first + last) / 2;
        if (a[mid] == target) { // found it!
            return mid;
        } else if (a[mid] < target) { // middle element too small; search right half
            return binarySearch(a, target, mid+1, last);
        } else { // a[mid] > target
            // middle element too large; search left half
            return binarySearch(a, target, first, mid-1);
        }
    }
}
```

### Traversing a Maze

- Show java code in eclipse