Strings

Problem 2

Ask the user for a sentence (one string). Take each word and reverse the order of its letters, keeping the order of the words unchanged. Output the modified sentence (one string). Assume that the sentence is properly capitalized and ends with a period, and do the same to the output. You may assume that the input does not contain digits, punctuation, or extraneous spaces.

Example:

java ReverseWords

The quick brown fox jumps over the lazy dog.

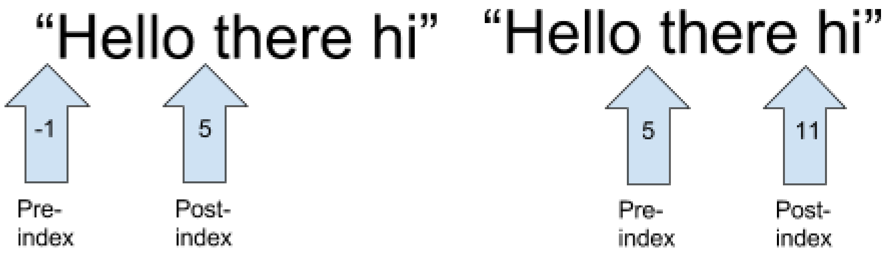
RESULT: "Eht kciuq nworb xof spmuj revo eht yzal god."

Code: See Problem2.java

Explanation

Option 1

Keep track of all of the spaces with a pre- and post- Index tracker. Keep moving both indexes up as we go through each word.



Option 2

Use String.split() to return an array of Strings that are split by the parameter

Example:

"Hello there hi" with parameter of " " (line 28) becomes an array containing:

Hello	there	hi

Recursion

Problem 4

Write a method that takes a string and produces a backwards version of it. (for example, "good gravy" backwards is "yvarg doog").

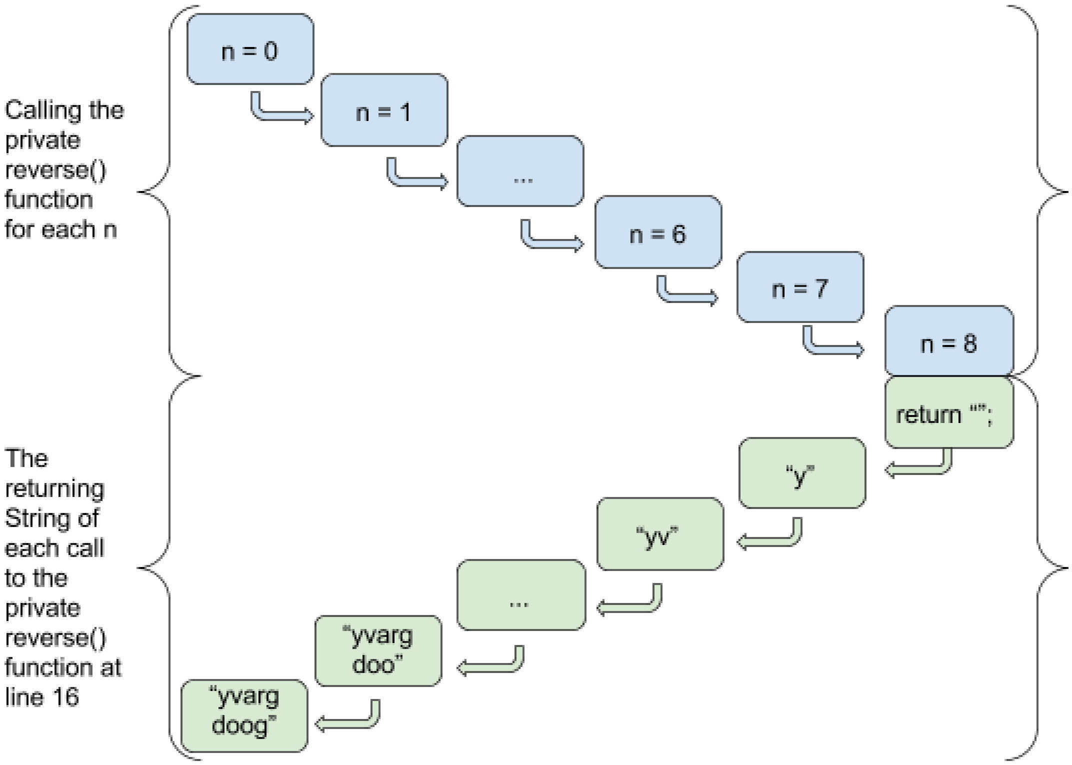
Code See Problem4.java

Explanation

Variables:

- str → the string to reverse
- n → the index of the character

Go through each index of the string and print out each letter at index n, starting from the back of the string to the front (aka go from index str.length()-1 to index 0). Take each character and append it to the end of the resulting string.


Calling the private reverse() function for each n
n = 0
n = 1
...
n = 6
n = 7
n = 8
return "";
"y"
"yv"
...
"yvarg doo"
"yvarg doog"
The returning String of each call to the private reverse() function at line 16

Efficiency Analysis

Problem 8.a

```
double[] combine(double[] array1, double[] array2){
        double[] result = new double[array1.length +
                                        array2.length];
        int r = 0;

        for (int i = 0 ; i < array1.length ; i++){
                result[r] = array1[i];
                r++;
        }
        for (int i = 0 ; i < array2.length ; i++){
                result[r] = array2[i];
                r++;
        }
        return result;
}
```

Variables

- n = array1.length
- m = array2.length

Operations

1. (1 work) Each for-loop iteration, which includes
    a. Reading from array1 or array2
    b. Writing to result
    c. Increasing i and r
    d. For-loop conditional check
2. (1 work) Creating an array

Work Count

- 1 → creating an array once
- n → doing operation 1 for array1.length many times
- m → doing operation 1 for array2.length many times

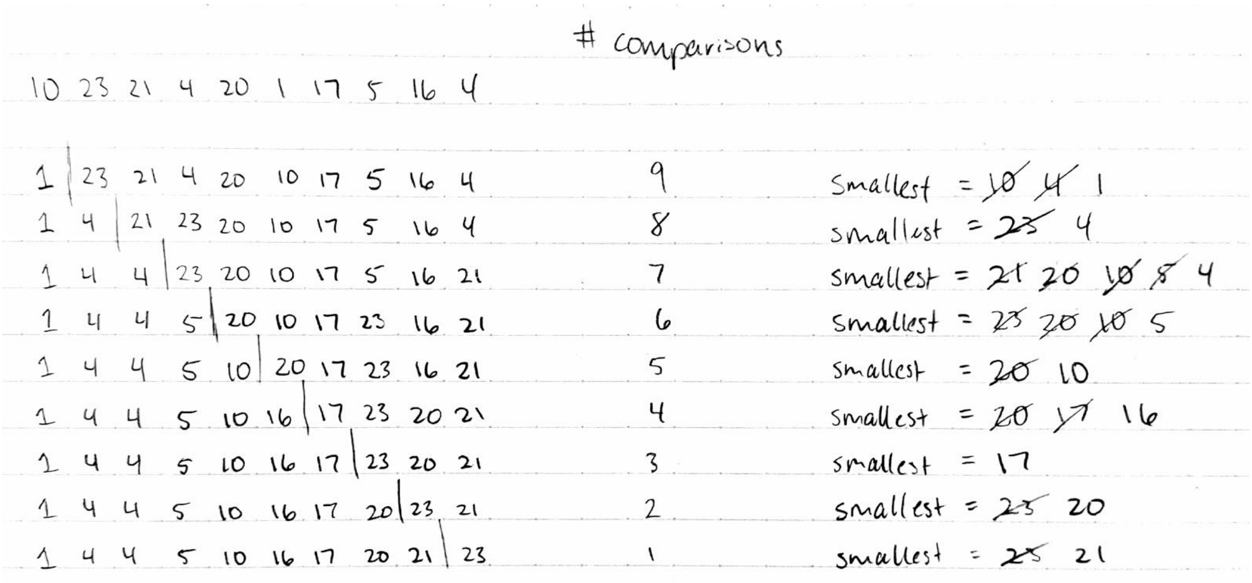
Big-O

$O(1) + O(n) + O(m) = O(1 + n + m) = O(n + m)$
** O(1) is dropped because it's insignificant in the long run (and because constants are dropped)

Searching/Sorting

Problem 9

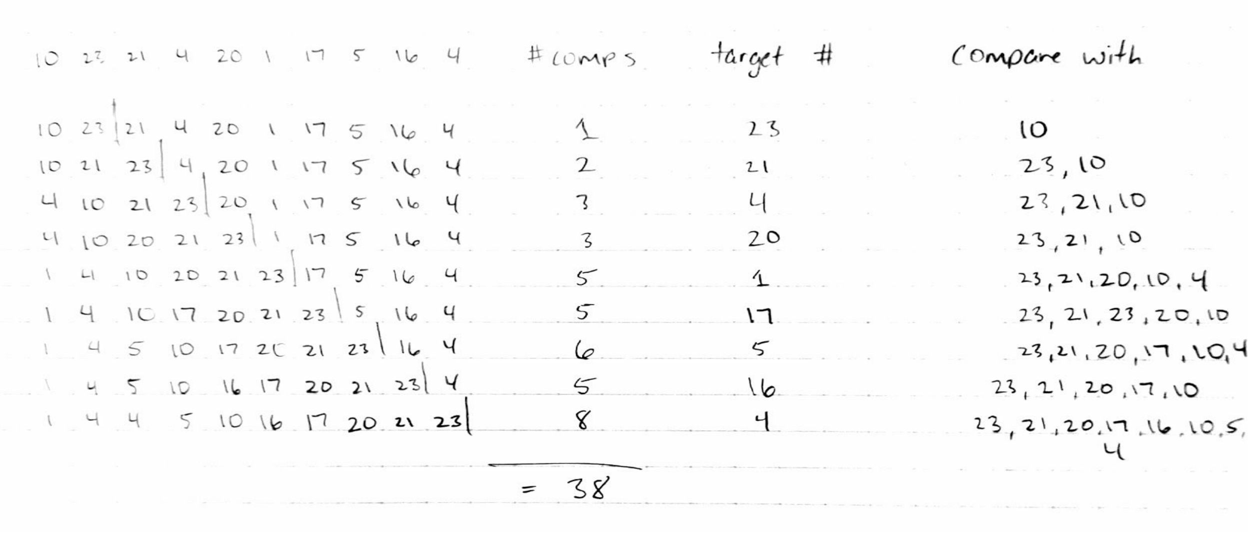
1. Trace selection sort on the following array of letters (sort into alphabetical order):
   a. 10, 23, 21, 4, 20, 1, 17, 5, 16, 4



	# comparisons	
10 23 21 4 20 1 17 5 16 4		
1 | 23 21 4 20 10 17 5 16 4	9	Smallest = ~~10~~ ~~4~~ 1
1 4 | 21 23 20 10 17 5 16 4	8	smallest = ~~23~~ 4
1 4 4 | 23 20 10 17 5 16 21	7	smallest = ~~21~~ ~~20~~ ~~10~~ ~~5~~ 4
1 4 4 5 | 20 10 17 23 16 21	6	smallest = ~~23~~ ~~20~~ ~~10~~ 5
1 4 4 5 10 | 20 17 23 16 21	5	smallest = ~~20~~ 10
1 4 4 5 10 16 | 17 23 20 21	4	smallest = ~~20~~ ~~17~~ 16
1 4 4 5 10 16 17 | 23 20 21	3	smallest = 17
1 4 4 5 10 16 17 20 | 23 21	2	smallest = ~~23~~ 20
1 4 4 5 10 16 17 20 21 | 23	1	smallest = ~~23~~ 21

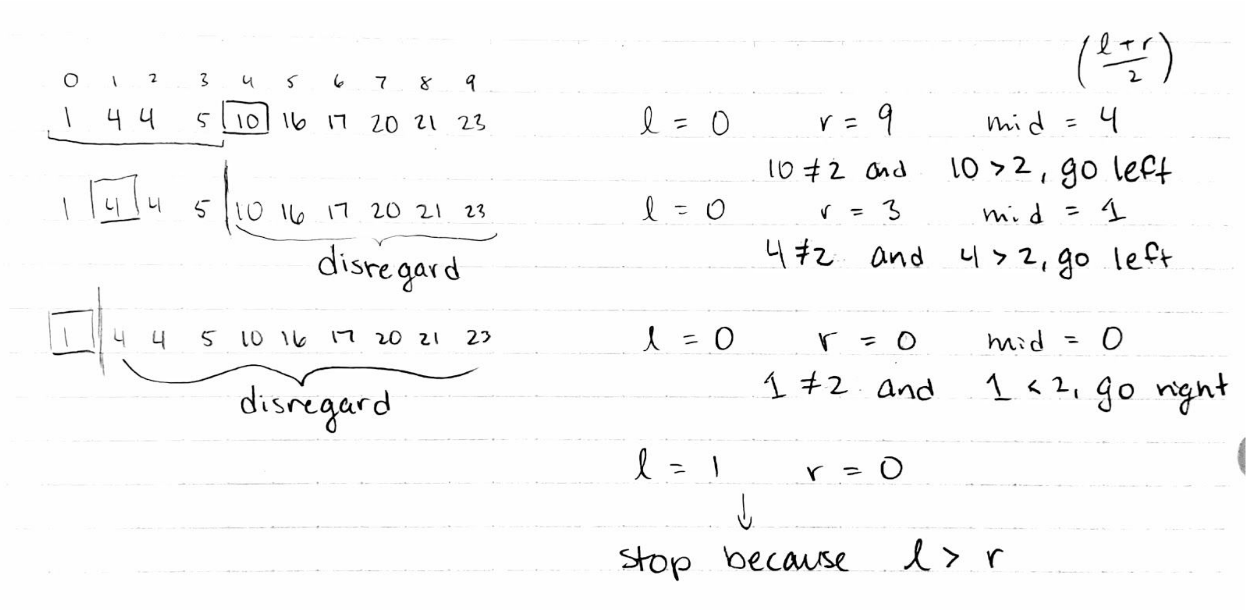
Total # of comparisons = 1 + 2 + 3 + … + 8 + 9 = $\frac{n(n+1)}{2}$

2. Trace insertion sort on the following array of letters (sort into alphabetical order):
   a. 10, 23, 21, 4, 20, 1, 17, 5, 16, 4



10 23 21 4 20 1 17 5 16 4	# comps	target #	Compare with
10 23 | 21 4 20 1 17 5 16 4	1	23	10
10 21 23 | 4 20 1 17 5 16 4	2	21	23, 10
4 10 21 23 | 20 1 17 5 16 4	3	4	23, 21, 10
4 10 20 21 23 | 1 17 5 16 4	3	20	23, 21, 10
1 4 10 20 21 23 | 17 5 16 4	5	1	23, 21, 20, 10, 4
1 4 10 17 20 21 23 | 5 16 4	5	17	23, 21, 23, 20, 10
1 4 5 10 17 20 21 23 | 16 4	6	5	23, 21, 20, 17, 10, 4
1 4 5 10 16 17 20 21 23 | 4	5	16	23, 21, 20, 17, 10
1 4 4 5 10 16 17 20 21 23 |	8	4	23, 21, 20, 17, 16, 10, 5, 4
	= 38		

3. Trace binary search for the **number 2** on the following array of letters:
    a. 1, 4, 4, 5, 10, 16, 17, 20, 21, 23



$\left(\frac{l+r}{2}\right)$

0 1 2 3 4 5 6 7 8 9

$\underbrace{1\ 4\ 4\ 5}\ \boxed{10}\ 16\ 17\ 20\ 21\ 23$ $l = 0$ $r = 9$ mid $= 4$

10 ≠ 2 and 10 > 2, go left

$1\ \boxed{4}\ 4\ 5\ |\ \underbrace{10\ 16\ 17\ 20\ 21\ 23}_{\text{disregard}}$ $l = 0$ $r = 3$ mid $= 1$

4 ≠ 2 and 4 > 2, go left

$\boxed{1}\ |\ \underbrace{4\ 4\ 5\ 10\ 16\ 17\ 20\ 21\ 23}_{\text{disregard}}$ $l = 0$ $r = 0$ mid $= 0$

1 ≠ 2 and 1 < 2, go right

$l = 1$ $r = 0$

↓

stop because $l > r$