Course: CSC 225, Fall 2013

If you want attendance credit, hand this in!

Name: Please print your name.

E-mail: And your e-mail address.

Please mark each square for which you CANNOT attend (with an X, or preferably put your class number in the box to aid in error detection).

Time/Day	Mon.	Tues.	Wed.	Thurs.	Fri.
9:30-10:00					
10:00-10:30					
10:30-11:00		CSC 225	CSC 225		CSC 225
11:00-11:30		CSC 225	CSC 225		CSC 225
11:30-12:00		CSC 320	CSC 320		CSC 320
12:00-12:30	CSC 225 Lab	CSC 320	CSC 320		CSC 320
12:30-1:00	CSC 225 Lab				
1:00-1:30	CSC 225 Lab				
1:30-2:00	CSC 225 Lab				
2:00-2:30					
2:30-3:00					CAG
3:00-3:30					CAG
3:30-4:00					

CSC 225 is done at 11:30 so I did NOT fill in the box for 11:30 for our class.

Lab 2 (for Monday Sept. 16) has been posted.
If you want extra problems for labs, let me know.

Assignment #1:
Written questions: due Fri. Sept. 20.
Programming questions: due Tues. Sept. 24 (connex).

CSC Revealed Clinic Friday Sept. 13, 2:30 in ECS 348:
Installing Java on your computer, and running and compiling your first program. All students are welcome to volunteer or participate.

Make sure you sign the attendance sheet when there is one.

Slide by: Kevin Wayne
Princeton University

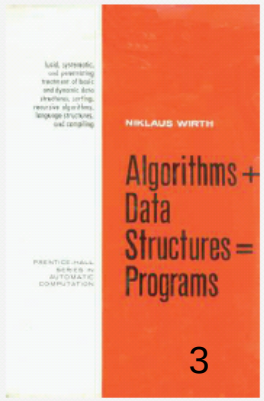
Why study algorithms?

To become a proficient programmer.

" I will, in fact, claim that the difference between a bad programmer and a good one is whether he considers his code or his data structures more important. Bad programmers worry about the code. Good programmers worry about data structures and their relationships. "

— Linus Torvalds (creator of Linux)

" Algorithms + Data Structures = Programs. " *— Niklaus Wirth*



To check your answer for question 4:

Add to top of readRear:

```
int count=0;
```

Inside the loop:

```
current= start;
while (current.next != null)
{
        current= current.next; // Statement to count.
        count++;
        // Checkpoint 2.
 }
```

At the end:

```
System.out.println("Executed " + count + " times on list of size " + n);
```

Divide and Conquer

1. Divide the problem into two or more subproblems.
2. Solve the subproblems.
3. Marry the solutions.

This is one of the most common problem solving tactics and leads naturally to recursive algorithms.

How can you design a divide and conquer algorithms for reversing a list?

Assignment: the divide step splits a list of size n into two lists, one with the first floor(n/2)items from the list and

the second one with the next ceiling(n/2) items.

To design a recursive algorithm: don't try to think deeper than one level. Just assume when you call your routine recursively that its behaviour matches the specifications for the routine in designing the top level.

Consider this recurrence which is only defined for values of $n = 2^k$ for some integer $k \geq 0$:

$T(1) = 1$, $T(n) = n + T(n/2)$.

(a) Solve this recurrence using repeated substitution.

(b) Prove the answer is correct using induction.

(c) Suppose the base case is changed to give a recurrence that is only defined for values of $n = 2^k$ for some integer $k \geq 2$:
$T(4) = 5$, $T(n) = n + T(n/2)$. What is the solution?