Homework 4 (quiz) due 10 pm Friday, October 7th
(one attempt -- 2 hrs to complete -- available by 10am Wednesday morning)

Program 1 Peer Reviews: each student has been assigned two submissions to review. Please comment on the readability of the code assigned to you. Did the submission follow the style and commenting guidelines? Rate the overall work on a scale of 1-10 and share your comments and suggestions with the student who submitted the work. Then, and read the feedback left for you. Complete Peer Review before next week Tuesday.

Program 2 due 10 pm Sunday, October 23rd

Late work is not Accepted

Last Time
Linked List Class
Linked List Variations
• header node
• tail reference
LinkedListIterator Class
Iterable and For-Each Loops

Today
Iterable and For-Each Loops (from last lecture)
More Linked List Variations
• double linking
• circular linking
Complexity
• concept
• big-O notation
• analyzing algorithms practice
• analyzing Java code
• practice analyzing Java code

Next Time
Read: finish Complexity
Complexity
• best/worst cases
• significance of scaling
• complexity caveats
Comparing ArrayList vs LinkedList
• shadow array - improving array resizing
Double and Circular Linking

Doubly-Linked Chains of Nodes (must define a new node class)

Circular Singly-Linked Chains of Nodes

Circular Doubly-Linked Chains of Nodes
Analyzing Algorithm Efficiency

Complexity

If problem size doubles and the number of operations:
Example: Complexity Analysis of Giving a Toast

What is the problem size?

What is the algorithm?

How can we characterize the problem?
Complexity Analysis:

- 

-
Big-O Notation

Concept

Some Growth Rate Functions in order of complexity

Simplifying Equations

Formal Definition
Complexity of Java Code

Basic operations

Sequence of statements

```
statement1;
statement2;
...
statementk;
```

If-else

```
if (cond) {
   //if sequence of statements
}
else {
   //else sequence of statements
}
```
Complexity of Java Code (cont.)

Basic loops
→ What is the problem size based on?

```java
for (i = 0; i < j; i++) {
    //sequence of statements
}
```

Nested loops
→ What is the problem size based on?

```java
for (i = 0; i < N; i++) {
    for (j = 0; j < M; j++) {
        //sequence of statements
    }
}
```

Loops with nested method calls (assume problem size based on N)

```java
for (i = 0; i < N; i++) {
    f1(i);  //assume O(1)
}
```

```java
for (i = 0; i < N; i++) {
    f2(N);  //assume O(N)
}
```

```java
for (i = 0; i < N; i++) {
    f3(i);  //assume O(i)
}
```
Practice - Complexity of Java Code

**method1**

→ What is the problem size based on?

```java
public void method1(int[] A) {
    for (int i = 0; i < A.length - 1; i++)
        method2(A, i);
}
```

**method2**

```java
public void method2(int[] B, int s) {
    for (int i = s; i < B.length - 1; i++)
        if (B[i] > B[i+1])
            method3(B, i, i+1);
}
```

**method3**

```java
public void method3(int[] C, int x, int y) {
    int temp = C[x];
    C[x] = A[y];
    C[y] = temp;
}
```
Practice - Complexity of Java Code

method4
What is the problem size based on?

```java
public void method4(int Q) {
    int sum = 0, R = 1000;

    for (int i = Q; i >= 1; i--)
        for (int j = 0; j < R; j++)
            sum += j;
}
```

method5
What is the problem size based on?

```java
public void method5(int X) {
    int tmp, arr[];

    arr = new int[X];
    for (int i = 0; i < X; i++)
        arr[i] = X - i;

    for (int i = 0; i < X - 1; i++) {
        for (int j = i; j < X - 2; j++) {
            if (arr[j] > arr[j+1]) {
                tmp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = tmp;
            }
        }
    }
}
```