M3: Hands-On: Linked Nodes

Due Feb 26 at 11:59pm **Points** 2 **Questions** 2 **Time Limit** None
Allowed Attempts Unlimited

Instructions

breakpoint.

4. Step in to the call to `basicExamples()` .
5. Single-step to line 52: `m = new Node(2);`
6. Open a viewer on `n` .
7. Step over each remaining statement in `basicExamples` , making sure you understand the effect of each statement. (You may want to step in to the calls to `length` and `contains` .)

Close the viewer and end program execution.

1. Clear any previous breakpoints you set in `DoublyLinked.java` .
2. Set a breakpoint on line 19: `client.add()` .
3. Start the debugger and wait until execution is paused at the breakpoint.
4. Step in to the call to `add()` .
5. Single-step to line 139: `System.out.println(toString(n));`
6. Open a viewer on `n` .
7. Click on the *Interactions* tab in the jGRASP Desktop.
8. Use *Interactions* to practice inserting the node referenced by `temp` at various locations in the pointer chain. (You'll have to repeat these steps each time you want to practice an insertion.)

Take the Quiz Again

Attempt History

	Attempt	Time	Score
KEPT	Attempt 2	less than 1 minute	2 out of 2
LATEST	Attempt 2	less than 1 minute	2 out of 2
	Attempt 1	9 minutes	1 out of 2

Score for this attempt: **2** out of 2
Submitted Feb 24 at 8:47pm
This attempt took less than 1 minute.

Question 1

1 / 1 pts

What *singly-linked* list of nodes is accessible from n after the following statements have executed?

```
Node n = new Node(1);
n.next = new Node(2, new Node(3));
n.next = n.next.next;
n = new Node(4, n);
n.next.next = new Node(5);
n.next = new Node(6, n.next);
```

A. [1]→[2]→[3]→[4]→[5]→[6]

B. [6]→[5]→[4]→[3]→[2]→[1]

C. [4]→[6]→[1]→[5]

D. [4]→[5]→[6]

- ○ A
- ○ B
- Correct! [✓ C]
- ○ D

Question 2

1 / 1 pts

What *doubly-linked* list of nodes is accessible from `n` after the following statements have executed?

```
Node n = new Node(1);
n.prev = new Node(2);
n.next = new Node(3);
n.prev.next = n;
n.next.prev = n;
n = n.prev;
Node m = n.next;
Node p = new Node(4);
p.prev = m;
p.next = m.next;
m.next = p;
p.next.prev = p;
m = null;
p = null;
```

A. [1]⇆[2]⇆[3]⇆[4]

B. [2]⇆[1]⇆[4]⇆[3]

C. [4]⇆[3]

D. [3]

○ A

Correct!

[✓ B]

○ C

○ D

Quiz Score: **2** out of 2