1. What will be the output?

```
class Node {
    int data;
    Node prev, next;
}

public class Main {
    public static void main(String[] args) {
        Node first = new Node();
        Node second = new Node();

        first.data = 10;
        second.data = 20;

        first.next = second;
        second.prev = first;

        System.out.println(second.prev.data);
    }
}
```

a) 10
b) 20
c) null
d) Compilation error

Answer: a) 10

Explanation: second.prev points to first, whose data is 10.

2. What does this print?

```
Node a = new Node();
Node b = new Node();
a.data = 5;
b.data = 15;

a.next = b;
b.prev = a;

System.out.println(a.next.data);
```

a) 5
b) 15
c) null
d) 0

Answer: b) 15

Explanation: a.next is b, and b.data is 15.

3. What will be printed?

```
Node n1 = new Node();
Node n2 = new Node();
Node n3 = new Node();

n1.data = 1; n2.data = 2; n3.data = 3;

n1.next = n2;
n2.prev = n1;
n2.next = n3;
n3.prev = n2;

System.out.println(n3.prev.prev.data);
```

a) 1
b) 2
c) 3
d) null

Answer: a) 1

Explanation: n3.prev → n2, n2.prev → n1, so data = 1.

4. Predict the output:

```
Node start = new Node();
start.data = 100;

start.next = new Node();
start.next.data = 200;
start.next.prev = start;

System.out.println(start.next.prev.data);
```

a) 100
b) 200
c) 0
d) null

Answer: a) 100

Explanation: start.next is 200, its prev is start, so prints 100.

5. Output of reverse traversal:

```
Node a = new Node();
Node b = new Node();
Node c = new Node();

a.data = 1; b.data = 2; c.data = 3;

a.next = b; b.prev = a;
b.next = c; c.prev = b;
```

```
Node current = c;
while (current != null) {
    System.out.print(current.data + " ");
    current = current.prev;
}
```

a) 1 2 3
b) 3 2 1
c) 1 3
d) Compilation error

Answer: b) 3 2 1

Explanation: Traversal from c to a via prev pointer.

6. What does this print?

```
Node head = new Node();
head.data = 5;
Node second = new Node();
second.data = 10;
head.next = second;
second.prev = head;
second.next = head;
head.prev = second;

System.out.println(head.prev.data);
```

a) 10
b) 5
c) null
d) Infinite loop

Answer: a) 10

Explanation: head.prev was reassigned to second.

7. What is the output here?

```
Node node = new Node();
node.data = 50;

node.next = node;
node.prev = node;

System.out.println(node.next.prev.data);
```

a) 50
b) null
c) Runtime error
d) Compilation error

Answer: a) 50

Explanation: Circular self-linking: node.next = node, so still accesses same node.

8. What happens in this reverse loop?

```
Node n1 = new Node(); n1.data = 1;
Node n2 = new Node(); n2.data = 2;
Node n3 = new Node(); n3.data = 3;
n1.next = n2; n2.prev = n1;
n2.next = n3; n3.prev = n2;
Node temp = n3;
while (temp != null) {
   System.out.print(temp.data + " ");
   temp = temp.prev;
}
```

a) 3 2 1
b) 1 2 3
c) null
d) 1 3

Answer: a) 3 2 1

Explanation: Standard reverse traversal using .prev.

9. What is the output of this circular doubly-linked setup?

```
Node a = new Node();
a.data = 1;
a.next = a;
a.prev = a;
System.out.println(a.prev.next.data);
```

a) 1
b) 0
c) null
d) Compilation error

Answer: a) 1

Explanation: Both prev and next point to a, so result is 1.

10. Which statement is true about the output of this code?

```
Node a = new Node(); a.data = 1;
Node b = new Node(); b.data = 2;
a.next = b;
b.prev = a;
System.out.println(a.prev);
```

a) Prints 1
b) null
c) Runtime error
d) Compilation error

Answer: b) null

Explanation: a.prev is never initialized, so defaults to null.