1 Forget It, We Ball

The 61Ballers are organizing the best IM team at Cal, but they first need your help with some inheritance issues...

Suppose we have the Person interface and the Athlete, and SoccerPlayer classes defined below.

```java
interface Person {
    void speakTo(Person other);
    default void watch(Athlete other) { System.out.println("wow"); }
}

public class Athlete implements Person {
    @Override
    public void speakTo(Person other) { System.out.println("i love sports"); }
    @Override
    public void watch(Athlete other) { System.out.println("ball is life"); }
}

public class SoccerPlayer extends Athlete {
    @Override
    void speakTo(Person other) { System.out.println("join 61ballers"); }
}
```

Read the code below and fill in the table on the next page.

For lines 1-11, write down the static type of the object being created in the "Compile Time (Static)" column, the dynamic type in the "Runtime (Dynamic)" column. For the output, write nothing if there are no errors, write CE if there's a compiler error, and write RE if there's a runtime error.

For lines 13-25, identify the method that's been saved during compile time, and write down its name and the class it belongs to in the "Compile Time (Static)" column. Identify the method executed at runtime, and write down its information in the "Runtime (Dynamic)" column. Write output in the "Output" column, if anything. Write CE if there is a compiler error and RE if there is a runtime error. If a line errors, continue executing the rest of the lines.

```java
Person ayati = new Person();

Athlete aniruth = new SoccerPlayer();

SoccerPlayer vanessa = aniruth;

Person eric = new Athlete();

Athlete shreyas = new Athlete();
```

```
10
11   SoccerPlayer yaofu = new SoccerPlayer();
12
13   eric.watch(aniruth);
14
15   shreyas.speakTo(yaofu);
16
17   yaofu.speakTo(eric);
18
19   ((Athlete) yaofu).speakTo(eric);
20
21   ((Person) yaofu).speakTo(eric);
22
23   ((Athlete) eric).speakTo(shreyas);
24
25   ((SoccerPlayer) eric).watch(yaofu);
```

Line	Compile Time (Static)	Runtime (Dynamic)	Output
1			
3			
5			
7			
9			
11			
13			
15			
17			
19			
21			
23			
25			

2   List Inheritance

Modify the code below so that the max method of DMSList works properly. Assume all numbers inserted into DMSList are positive, and we only insert using `insertFront`. You may not change anything in the given code. You may only fill in blanks. You may not need all blanks. (Spring '16, MT1)

```java
public class DMSList {
    private IntNode sentinel;

    public DMSList() {
        sentinel = new IntNode(-1000, _____);
    }

    public class IntNode {
        public int item;
        public IntNode next;
        public IntNode(int i, IntNode h) {
            item = i;
            next = h;
        }

        public int max() {
            return Math.max(item, next.max());
        }
    }

    class LastIntNode extends IntNode {
        public LastIntNode() {

            _____;
        }

        @Override
        public int max() {

            _____;
        }
    }

    /* Returns 0 if list is empty. Otherwise, returns the max element. */
    public int max() {
        return sentinel.next.max();
    }

    public void insertFront(int x) { sentinel.next = new IntNode(x, sentinel.next); }
}
```