Instructions: Use the space provided for each answer. If necessary, use the back of the page. You must complete this exam independently, with no outside resources of any kind. This exam consists of 5 questions, worth a total of 100 points.

Below, we provide the Iterator and (partial) List interfaces, should you need them for reference:

interface Iterator<E>:

// Returns true if the iteration has more elements.
boolean hasNext();

// Returns the next element in the iteration.
E next();

// Removes from the underlying collection the last element returned by the iterator.
void remove();

interface List<E>:

boolean add(E e); // add to the end of the list
void add(int index, E element); // insert at specified index
void clear(); // remove all elements
boolean contains(Object o); // true if list contains the object
E get(int index); // get element at specified index
int indexOf(Object o); // index of first occurrence (or -1)
boolean isEmpty(); // list is empty (size() == 0)
Iterator<E> iterator(); // return iterator though this list
int lastIndexOf(Object o); // index of last occurrence (or -1)
E remove(int index); // remove at specified index
boolean remove(Object o); // remove first occurrence of object
E set(int index, E element); // set item at index to be given element
int size(); // number of elements in list
1. (10 points)
Suppose the following class has been defined:

```java
public class IntRef {
    public int data;

    public IntRef(int data) {
        this.data = data;
    }
}
```

What does the following code print out when run?

```java
IntRef var1 = new IntRef(1);
IntRef var2 = new IntRef(2);
IntRef var3 = var1;
IntRef var4 = var2;

var3.data = 3;
var4.data = 4;
var1 = var4;
var2 = var1;
var4 = var3;
var2.data = 5;
var4.data = 6;
var1.data = 7;

System.out.println("1: " + var1.data);
System.out.println("2: " + var2.data);
System.out.println("3: " + var3.data);
System.out.println("4: " + var4.data);
```
2. (10 points) What does the following code print out for each input? (Recall that the String representation of a list is comma-separated and surrounded by brackets, as in the specified inputs below.)

```java
public static void mystery(List<Integer> l)
{
    List<Integer> list = new LinkedList<Integer>();
    Iterator<Integer> iter = l.iterator();
    while (iter.hasNext()) {
        if (iter.next() % 2 == 0) {
            list.add(0, iter.next());
        } else {
            list.add(iter.next());
        }
    }
    System.out.println(list);
}
```

[1,2,3,4]  

[4,3,2,1]  

[1,1,2,3,5,8,13,21]
3. (20 points) Use the following class definitions to answer the questions on the next page. (You may tear out this page if helpful.)

```java
public class Sheep {
    public void m1() {
        System.out.println("S1");
    }
}

public class Baa extends Sheep {
    public void m3() {
        System.out.println("B3");
        m1();
    }
}

public class Ram extends Baa {
    public void m2() {
        m3();
        System.out.println("R2");
    }
    public void m3() {
        super.m3();
        System.out.println("R3");
    }
}

public class Ewe extends Ram {
    public void m1() {
        System.out.println("E1");
        m2();
    }
    public void m3() {
        System.out.println("E3");
    }
}
```
Suppose the following variables are defined:

```java
Sheep s1 = new Baa();
Ram r1 = new Ram();
Ram r2 = new Ewe();
```

Indicate on each line below the output produced by each statement shown. If the statement produces more than one line of output indicate the line breaks with slashes as in a/b/c to indicate three lines of output with a followed by b followed by c. If the statement causes an error (either at compile time or runtime), write the word “error” to indicate this.

```java
s1.m1();
-----------------------------------------------
s1.m2();
-----------------------------------------------
r1.m3();
-----------------------------------------------
r2.m2();
-----------------------------------------------
((Ram) s1).m1();
-----------------------------------------------
```

(Problem instructions by Reges and Stepp.)
4. (20 points) Add the method `keepLast(int k)` to `LinkedIntList`, which removes all but the last `k` nodes. If the list has fewer than `k` nodes, then the list is unchanged. For example, if the `LinkedIntList` contains `[1, 2, 3, 4, 5]`, then after calling `keepLast(2)`, it should contain `[4, 5]`. **You may not call any other methods of `LinkedIntList`**. You may not create any new `ListNode` variables.

```java
public class ListNode {
    public int data;
    public ListNode next;
}

public class LinkedIntList {
    // Reference to the first node in the list
    private ListNode front;

    ...
}
```

Below is the declaration of the method you are to implement as a member of the `LinkedIntList` class. (If you run out of space, you are free to use the back of the page or ask for scratch paper.) **Remember not to call any other methods in `LinkedIntList`**.

```java
/**
 * Keep only the last k elements in the list.
 * @param k number of numbers to keep
 */
public void keepLast(int k)
```
5. (20 points) On the new “MyFace” social network site, there are several kinds of pages, each with a title and URL. Pages are represented by the Page class, which has the following public methods (and no public or protected member variables):

- Page(String title, String url)
  Constructs a Page with the specified title and URL.
- String title()
  Returns the Page’s title.
- String url()
  Returns the Page’s URL.

Your task is to design the new ProfilePage class, which is a special kind of Page that has a first name, last name, and list of links to other pages. The title of a ProfilePage should be the first and last name, separated by a space (e.g., “Barack Obama” or “Hillary Clinton”).

In addition to the methods inherited from Page, your class must include the following new methods:

- ProfilePage(String first, String last, String url)
  Constructs a ProfilePage with the specified first name, last name, and URL.
- void addLink(Page p)
  Add p to this ProfilePage’s list of linked pages. (Links do not have to be symmetric. In other words, Page p does not have to have a link back to this page.)
- List<Page> getLinks()
  Return a list of all pages linked from this page.
(Blank page you may use for solving Problem 5.)