a) (10 marks) Consider the following diagram of an **IntNode**-based linked list:

```
   head
    v
  4 -> 7 -> 5 -> 2 -> 9
```

Execute and give the printed output of the following **Java** code-fragment that calls recursive method **funky** as it operates on this linked list.

In main method:

```java
    IntNode cursor, a, b, head;
    a = head; b = head.getLink();
    funky(a, b, 1);
```

method **funky**:

```java
    public static void funky(IntNode a, IntNode b, int i){
        if (b.getLink() == null)
            a.setData(b.getData());
        else {
          System.out.println(i + ": ": + a.getData() + " " + b.getData());
          b.setData(a.getData());
          funky(a.getLink(), b.getLink(), i + 1);
          System.out.println(i + ": ": + a.getData() + " " + b.getData());
        }
    }
```

Answer:

List before call to **funky()**:

```
head -> 4 [a] -> 7 [b] -> 5 -> 2 -> 9
```
Output printed during execution of method funky():

1: 4 7
2: 4 5
3: 4 2
3: 4 9
2: 4 4
1: 4 4

List after call to funky():

head -> 4 [a] -> 4 [b] -> 4 -> 9 -> 9

b) (10 marks) Write the body of a recursive Java method

```java
public static int altSum(IntNode head, int i)
```

which, when called as altSum(head, 1), computes the alternating sum of the elements in the IntNode-based linked-list with head-reference head, i.e., \( \sum_{i=1}^{n} (sign(i) \times val(i)) \) where \( n \) is the number of nodes in the list, \( val(i) \) is the value associated with the \( i \)th element in the list, and \( sign(i) \) is 1 if \( i \) is odd and -1 if \( i \) is even. For example, given the list

```
head
```

![Linked List Diagram]

the alternating sum for this list is \( 4 + -7 + 8 + -2 + 9 = 12 \).

Answer:

```java
public static int altSum(IntNode head, int i){
    int sign;
    if (head == null)
        return(0);
    else {
        if ((i % 2) == 1)
            sign = 1;
        else
            sign = -1;
        return((sign * head.getData()) +
                altSum(head.getLink(), i + 1));
    }
}
```