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

```
head
```

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

\[
\text{public static int altSum(IntNode head, int i)}
\]

which, when called as \text{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

\begin{center}
\begin{tikzpicture}
    \node (o) at (0,0) {\textit{head}};
    \node (a) at (1,0) {$4$};
    \node (b) at (2,0) {$7$};
    \node (c) at (3,0) {$8$};
    \node (d) at (4,0) {$2$};
    \node (e) at (5,0) {$9$};
    \draw (o) -- (a);
    \draw (a) -- (b);
    \draw (b) -- (c);
    \draw (c) -- (d);
    \draw (d) -- (e);
\end{tikzpicture}
\end{center}

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