In this quiz, we will use and extend the class IntNode described in the textbook and lectures.

a) (12 marks) Consider the following diagram of an IntNode-based linked list starting at IntNode-reference head:

![Linked List Diagram]

Give the output of the following Java code-fragment as it operates on this linked list, and draw a diagram of the resulting linked list starting at head following the execution of this code-fragment.

```java
IntNode cursor, cursor1, cursor2, head;
int val, turn;

cursor1 = cursor2 = head;
turn = 2;
while (cursor2.getLink().getLink() != null) {
    cursor2 = cursor1.getLink();
    val = cursor1.getData();
    while (cursor2.getLink() != null) {
        val += cursor2.getData();
        cursor2 = cursor2.getLink();
    }
    if ((turn % 2) == 0)
        val = -val;
    System.out.println("(T,V) = (" + turn + "," + val + ")");
    cursor2 = new IntNode(val, cursor1.getLink().getLink());
    cursor1.setLink(cursor2);
    cursor2 = cursor1;
    cursor1 = cursor1.getLink();
turn++;
}
```
Answer:

Output:

\[(T, V) = (2, -18)\]
\[(T, V) = (3, -12)\]
\[(T, V) = (4, 8)\]
\[(T, V) = (5, 8)\]

Diagram:

```
head

5 -> 18 -> 12 -> 8 -> X
```

b) (8 marks) Write a two-parameter Java method `stutter` which, given the starting `IntNode reference head` of a linked list and an integer value `target`, counts the number of times `numOcc` that this target-value occurs in the list and adds `numOcc` nodes with value `target` to the end of that list. Note that this method does not return any value, i.e., it has return-type `void`.

Answer:

```java
public static void stutter(IntNode head, int target){
    IntNode temp;
    int i, numOcc = 0;

    if (head != null) {
        temp = head;
        while (temp.getLink() != null) {
            if (temp.getData() == target)
                numOcc++;
            temp = temp.getLink();
        }
        if (temp.getData() == target)
            numOcc++;
        for (i = 1; i <= numOcc; i++)
            temp.addNodeAfter(target);
    }
}
```