2) (10 pts) ALG (Linked Lists)

Consider the following code:

```
void doTheThing(node *head, node *current)
{
  if (current == NULL)
    return;

  else if (current == head->next)
  {
    if (current->data == head->next->next->data)
      doTheThing(head, head->next->next->next);
    else if (current->data == head->next->next->data + 1)
      doTheThing(head, head->next->next->next->next);
    else if (current->data == head->next->next->data + 5)
      doTheThing(head, current->next->next->next);
    else if (current->data == head->next->next->data + 10)
      doTheThing(head, head->next);
    else
      doTheThing(head, current->next);
  }

  else
    doTheThing(head, current->next);
}
```

Draw a linked list that simultaneously satisfies **both** of the following properties:

1. The linked list has **exactly four nodes**. Be sure to indicate the integer value contained in each node.

2. If the linked list were passed to the function above, the program would either crash with a segmentation fault, get stuck in an infinite loop, or crash as a result of a stack overflow (infinite recursion).

Note: When this function is first called, the head of your linked list will be passed as *both* arguments to the function, like so:

```
doTheThing(head, head);
```

Hint: Notice that all the recursive calls always pass *head* as the first parameter. So, within this function, *head* will always refer to the actual head of the linked list. The second parameter is the only one that ever changes.

Solution:

The only way to get wrecked with this code is to trigger the `doTheThing(head, head->next)` call. Since we only trigger that call when `current == head->next`, then making that recursive call results in infinite recursion. (Continued on the following page.)