

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.

3) (5 pts) ALG (Stacks and Queues)

Consider the following function:

```
void doTheThing(void)
{
    int i, n = 9; // Note: There are 9 elements in the following array.
    int array[] = {3, 18, 58, 23, 12, 31, 19, 26, 3};

    Stack *s1 = createStack();
    Stack *s2 = createStack();
    Queue *q = createQueue();

    for (i = 0; i < n; i++)
        push(s1, array[i]);

    while (!isEmptyStack(s1))
    {
        while (!isEmptyStack(s1))
            enqueue(q, pop(s1)); // pop element from s1 and enqueue it in q
        while (!isEmptyQueue(q))
            push(s2, dequeue(q)); // dequeue from q and push onto s2

        printf("%d ", pop(s2)); // pop from s2 and print element

        while (!isEmptyStack(s2))
            push(s1, pop(s2)); // pop from s2 and push onto s1
    }
    printf("Tada!\n");

    freeStack(s1);
    freeStack(s2);
    freeQueue(q);
}
```

What will be the exact output of the function above? (You may assume the existence of all functions written in the code, such as *createStack()*, *createQueue()*, *push()*, *pop()*, and so on.)