<u>Some more hints on the Midterm Assignment 2:</u>

Creating the reverse_circle should be fairly easy. After getting n, you can use a loop to generate numbers n, n-1, n-2, …, 2,1. After generating each number call the create_soldier function that returns you a dynamically allocated node. Then add that node to your linked list's rear position. In this process, maintaining a tail or rear node like queue can help as you will not need to traverse to the end every time. During this process tail's next will be head.

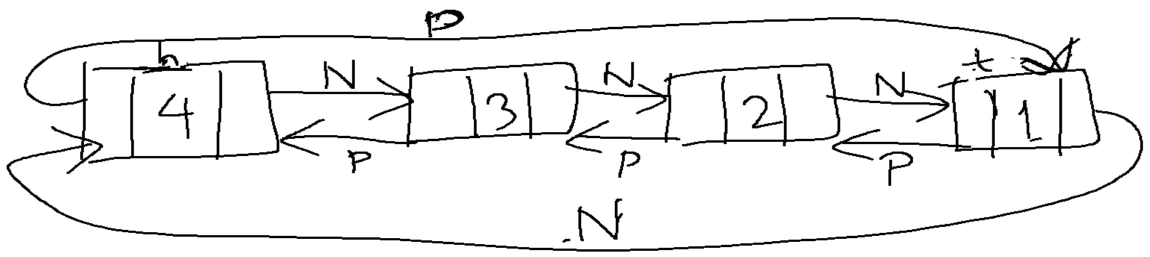
How would you know whether your linked list was created properly or not?

You can create a display() function and call it with the head of the linked list to print it. However, an usual printing of linked list might not give you a better idea whether your doubly properties of the linked list is correct. To check all the prev pointers is properly managed, you can create another function called display_from_back(). In this function you can start printing your linked list from tail and traverse from backward to head using prev pointer like you do for next pointer.

The above part is the base of further processing.

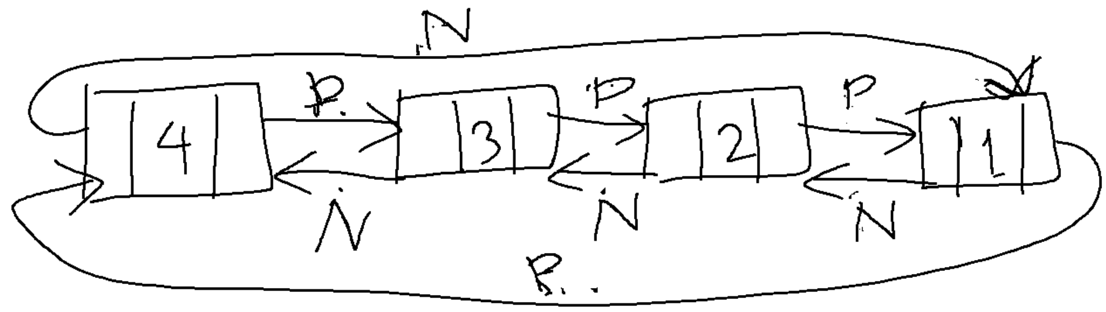
The re-arrange function: This function can be quite tricky.

Let's say you have 4 nodes. In that case your current linked list is like this. N = next pointer, P= previous pointer, h = head, and t = tail



Now, your target is to reverse the list, so that head will be the node with 1. And the each node's next and prev pointer will change the direction.

It is something like this you want to achieve.



See in the above picture, N became Prev, and prev became next. Isn't it kind of swapping problem? Yes, you just swap the next and prev of each node!

Let’s start the process. Let us use a node pointer cur for traversing. cur starts at head. As we know, for swapping between variables, we need a temp variable. So, we will need a temp node pointer.

cur is now at head. cur’s previous is the node with 1. However, we want cur’s previous should be node with 3 (which is actually cur’s next). So, we need to make cur’s next as cur’s prev.

if you manage to swap them properly, 4’s next will be 1 and 4’s prev will be 3.

Note that, we have not change anything for 3 yet.

So, our next node is 3. How can you go to 3 now?

3 is not in cur’s next anymore. 3 is actually cur’s prev. So, update cur accordingly and cur will be at 3.

Now, we have to follow the same process so that cur’s next become 4 and prev become 2 and so on……

In case of doubly linked list, do-while loop is sometimes a good choice as your cur starts at head and you will also stop when your cur becomes head after traversing.

Now, after completing the full traversal, you need to update your head.

See from the picture above, who should be your new head?

Now, you have the solders in the correct sequence and you will call the kill function.

Inside the kill function you can have a loop. Start from the head and skip k-1 nodes as delete the next.

There can be multiple way to track whether you have only one node. One easiest way would be keeping a counter variable. As soon as you delete one node, decrese counter. Befor the next skipping process, check counter already became 1 or not.

One important note. In this problem k has to be greater than 1. So, we will not use any test case with k<2.

That’s a lot of hints.

Good luck!