Linked Lists Practice

FOR ALL PROGRAMMIMNG PROBLEMS YOU MUST BE ABLE TO ALSO DRAW THE LISTS AND THE ACTIONS AS SHOWN IN CLASS.

Good practice problems, but NOT covered in the exam:

P1. `nodePT deleteKth(nodePT L, int k)` remove and FREE from the list L, the k-th node. The function will ***return* the list header** (the address of the first node in the list). E.g. if a list, L, has 5->8->1->9->20->7 , `deleteKth(L, 1)` will remove and free node 5 and return the address of the node 8. If the list has less than k items, no node is removed. Adjust the links. Do not copy the data content.

P2. `nodePT insertSorted(nodePT L, int val)` - insert a value in a SORTED list, L. . E.g. if L is 5->8->8->8->10->20->27, after `insertSorted(L, 15)` , L will be 5->8->8->8->10->15->20->27. Adjust the links. Do not copy the data content.

P3. `nodePT insertKth(nodePT L, int k, int val)` – insert in list L, at position k, a new node that will have value val. Assume positions are numbered from 1 so when k is 1, you insert at the beginning of the list. E.g. if L is 5->8->1->8->20->7, after `insertKth(L, 1,87)` , L will be 87->5->8->1->8->20->7 . If k is larger than the number of nodes in the list, insert at the end of the list. Adjust the links. Do not copy the data content.

P4. Write a function that takes as argument a node `p`, and swaps the two nodes following `p` *if the data in the first one is larger than that of the second one*. **You must readjust the links, not copy the data from one node into the other. No credit given otherwise.**

For example if p -> **6** -> **3** -> 2 -> … the two nodes following `p` have data 6 and 3 and since 6 >3 the nodes will be swapped (keep in mind that you must readjust the links, not just swap the values 3 and 6). If `p` -> **3** -> **6** -> 2 -> … the function will not swap: `p` -> **3** -> **6** -> 2 -> …

Assume the class provided representation of nodes and links:

```
typedef struct node * nodePT;

struct node  {
   int data;
   nodePT next;
};
```

a) The function does not crash (for pointer errors or otherwise). These points are only given if the program is also correct.

b) Draw a picture of what happens with the links when you swap the nodes. Use **line numbers (or code segments) to indicate on the picture** what line of your code produces those changes.

c) Write the function (Do not use anything that would bypass working with the links.)

P5. Write a function **`int triples(int* A, int N)`** that takes as argument an array, `A`, with `N` integers, and returns 1 if all the numbers in A appear a multiple of three times. (That is the same number could appear 3 times or 6 times or 21 times, etc.) Otherwise it will return 0.

You can assume that all the numbers in A are positive (greater or equal to 0).

E.g.: both `triples([5,3,5,3,3,5], 6)` and `triples([5,3,5,3,3,5,5,5,5], 9)` return 1. But `triples([3,7,3,3,7], 5)` returns 0 (7 appears only twice).

a) Give **both** the **time and space complexity** of your program. Justify it by referring back to specific program lines or putting comments in the program.

b) Give a brief but **clear** explanation of how your function works.

c) Write the code. Do all the data manipulation that is needed (if you want to use a specific algorithm, you need to write the code for it).

P6. Write a function that takes **the first nodes** of two lists (A and B) and checks if ***each*** $B_i = A_1 + A_2 + \ldots + A_i$, where A_1 and B_1 denote the first nodes from the lists. If yes, it returns 1, else it returns 0.
For
A: 3 -> 6 -> 2 -> 5 -> 13 ->1
B: 3 -> 9 -> 11 -> 16 -> 29 -> 30
It returns 1because: $B_1 = A_1 = 3$, $B_2 = A_1 + A_2$ (9 = 3+6),… , $B_6 = A_1 + A_2 + \ldots + A_6$ (30 = 3+6+2+5+13+1)

For
A: **3 -> 6 -> 2** -> 5 -> 13 ->1
B: 3 -> 9 -> **15** -> 16 -> 29 -> 30
It returns 0 because one or more nodes fail the property. In particular, $B_3 \neq A_1 + A_2 + A_3$ (15≠3+9+2),

a) Write the function. (you do **not** need to handle special cases). You should solve it using lists, not by copying the data in arrays and continuing to work with arrays. If you work with arrays, you lose 6 points.
Assume the class provided representation of nodes:

```
typedef struct node * nodePT;
```

```
struct node  {
   int data;
   nodePT next;
};
```

b) If your function fails or crashes for certain inputs, give **those inputs** and clearly **indicate the line with the problem** (use line numbers or write the test cases as a comment on that line of code).

c) What is the Θ complexity of your function? Justify your answer.

P7. Graded on correctness, little or no partial credit.

a) Write a function, int check(nodePT first_node), that takes as argument the first **node** of a **single linked list**. It should return 0 if all the items in the list are unique and 1otherwise (if there are repetitions). For example:

For 7-> 4 -> 9 -> 6 -> 4 -> 3   it returns 1 (4 is repeated)

For 5-> 2 -> 9 -> 6 -> 3 it returns 0 (no repetitions).

For a list with only one node it returns 0 (no repetitions).

Assume that links are implemented using the type and struct given below.

```
typedef struct node * nodePT;

struct node  {
   int data;
   nodePT next;
};
```

b) Give the time complexity of your function in terms of Θ.

P5. Write a function, **int my_count(nodePT L),** that takes as argument a list L of integers (the item is an integer). It should count the number of consecutive repetitions of each item and print both the item and the count. A single occurrence is counted as 0. The function should also ***return the total number of repetitions***. For example, for the list

7-> 2 -> 9 -> 9 -> 9 -> 9 -> 7 -> 7-> 9 -> 3 -> 3   it will print:

7, 0
2, 0
9, 3
7, 1
9, 0
3, 1

And it will return 5.
Your code should not crash.

Assume that list are implemented using dummy nodes. You should ***do the pointer manipulation*** (do not assume for example that there is a function that removes a node or one that inserts at a specific position).

```
typedef struct node * nodePT;
struct node  {
   int data;
   nodePT next;
};
```