## Lab 1 - Simply Linked Lists

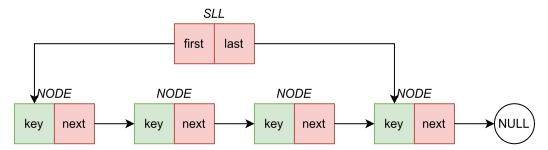
Implement a simply linked list using a structure similar to the following example:

```
typedef struct _NODE{
   int key;
   struct _NODE* next;
} NODE;

typedef struct {
   NODE* first;
   // pointer to last node is optional
   NODE* last;
} SLL:
```

Implement a function for each of the following operations:

- 1. List initialisation (allocate memory for the list) ★
- 2. Insert first ★
- 3. Delete first ★
- 4. Insert last ★
- 5. Delete last ★
- 6. Get the number of elements in the list  $\star$
- 7. List deinitialisation (free the memory allocated for the nodes and the list)  $\bigstar$
- 8. Search element by key. If the element is found return a pointer to the node, otherwise return 0. ★★
- 9. Delete element by key. If an element with the given key is found in the list delete it. \*\*
- 10. Delete node at arbitrary index. (E.g.: remove the 5th node) ★★
- 11. Concatenate 2 lists ★★
- 12. Insert node based on an ordering rule (E.g.: node with key 3 will be placed after nodes 1, 2 and before nodes 4, 5)  $\bigstar \bigstar$
- 13. Reverse list  $\star\star\star$
- 14. Merge 2 ordered lists into a single ordered list  $\star\star\star$
- 15. Reverse the list without allocating any additional memory.  $\star\star\star\star$



**Note:** Leave a comment with the text PB1, PB2, ... PB10 above every function that implements the respective lab task. (upper case text, no space between the text and the problem number)