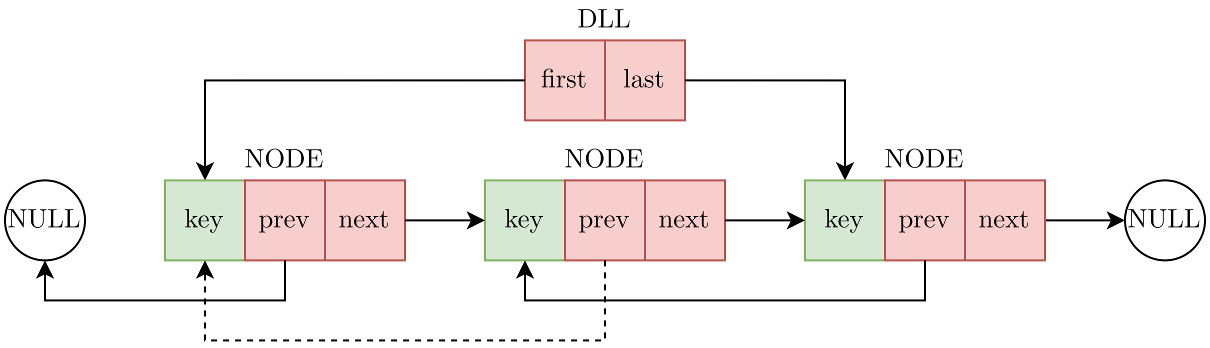
Lab 2 - Doubly Linked Lists

1 Doubly linked list

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

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

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



Implement a function for each of the following operations:

1. List initialisation (create an empty list) ★
2. Insert first ★
3. Delete first ★
4. Insert last ★
5. Delete last ★
6. Get the number of elements in the list ★
7. List deinitialisation (free the memory allocated for the nodes and the list) ★
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. Sort the list ★★

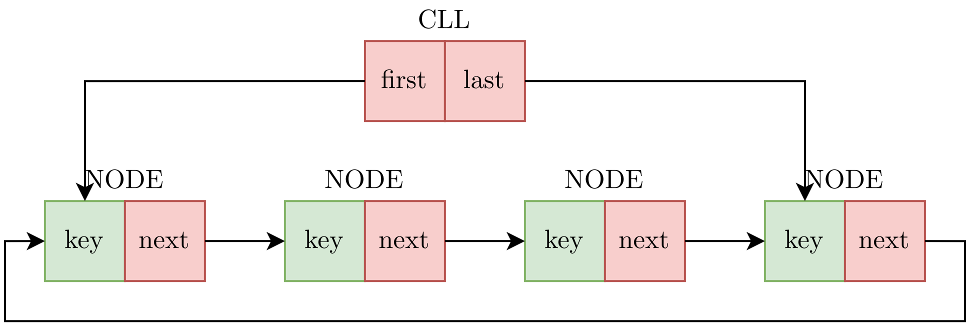
13. Insert node based on an ordering rule (E.g.: node with key 3 will be placed after nodes 1, 2 and before nodes 4, 5) ★★
14. Reverse list ★★★
15. Merge 2 ordered lists into a single ordered list ★★★
16. Reverse the list without allocating any additional memory. ★★★★

2 Circular linked list

Implement a circular 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;
} CLL;
```



Implement a function for each of the following operations:

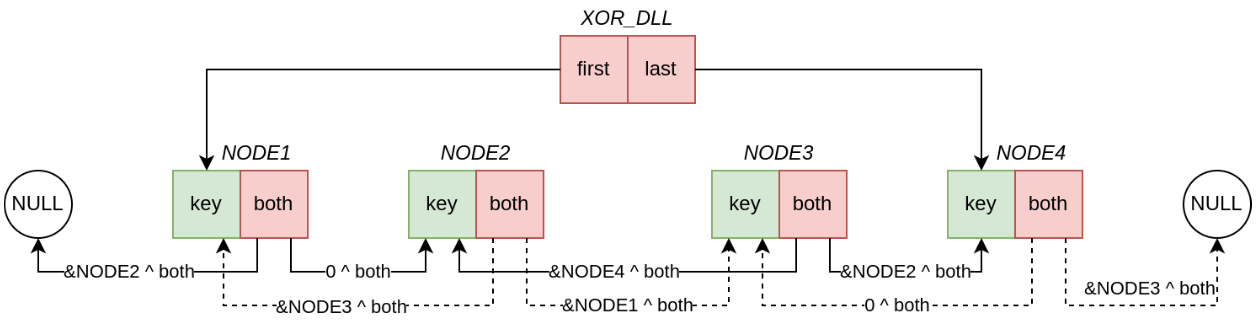
1. List initialisation (create an empty list) ★
2. Insert first ★
3. Delete first ★
4. Insert last ★
5. Delete last ★
6. List deinitialisation (free the memory allocated for the nodes and the list) ★

3 Xor doubly linked list

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

```
typedef struct _NODE{
    int key;
    size_t both;
} NODE;

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



Implement a function for each of the following operations:

1. List initialisation (create an empty list) ★
2. Insert first ★
3. Delete first ★
4. Insert last ★
5. Delete last ★
6. List deinitialisation (free the memory allocated for the nodes and the list) ★

Hard mode: Solve the lab problems using the containing record trick:

```
#define CONTAINING_RECORD(address, type, field) (\
    (type *)((char*)(address) - (size_t)(&((type *)0)->field)))
```

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)