Tip

- Through the whole course homework you must:
  - Compute time order
  - Compute memory order

Problem #1: Get back

- implemented int get_back()
- return value of last node
- find E..g. if list is 1 2 3 4 5 6
- get_back() should return node with value 6

Problem #1: Get front

- implemented int get_front()
- return value of last node
- find E..g. if list is 1 2 3 4 5 6
- Get_front() should return node with value 1

Problem #3: Delete front

- The opposite of insert front

```
LinkedList list;

list.insert_end(6);
list.insert_end(10);
list.insert_end(8);
list.insert_end(15);

list.delete_front();
list.print();
// 10 8 15
```

Problem #3: Delete back

- The opposite of insert back

```
LinkedList list;

list.insert_end(6);
list.insert_end(10);
list.insert_end(8);
list.insert_end(15);
```

list.delete_back()
//6 10 8

Problem #4: Get nth

- implemented int get_nth(int n)
- Given n-based position, return value of node number n
- find E..g. if list is 1 2 3 4 5 6
- get__back(3) should return node with value 3

Problem #4:search in Linked list

- implemented int find(int value)
- Given value , return if value in list or not
- If not in list return -1 else return first position for value
- find E..g. if list is 1 2 3 4 5 6
- get_nth_back(3) should return 2 (postion start from 0)