**Лабораторная работа №5**

Разработать классовую структуру, реализующую АТД список.

Четные варианты - односвязный список, нечетные - двусвязный.

Выполнять задание желательно в объектном стиле.

**Обязательные методы для всех:**

1. void add\_first(List, int value) Добавить новый элемент со значением value в начало
2. void delete\_first(List) Удалить первый элемент
3. void print(List) Распечатать список

Тестирование выполняется в отдельном классе.

В ходе тестирования нужно продемонстрировать работу всех реализованных методах в разных возможных ситуациях (включая, например, удаление из пустого списка и прочие подобные варианты).

**Номер варианта согласно номеру в журнале (важно соблюдать варианты).**

**Первой строкой в тестовом блоке программы вывести на консоль свой вариант и номера заданий согласно формуле.**

## **Номера заданий (v – номер варианта, n - количество заданий == 30):**

1. (v + 7) % n + 1
2. (v + 14) % n + 1
3. (v + 21) % n + 1
4. (v + 28) % n + 1
5. (v + 35) % n + 1
6. (v + 42) % n + 1

## **Задания:**

1. void print\_reversed(List) Распечатать список в обратном порядке
2. void add\_last(List, int value) Добавить новый элемент со значением value в конец
3. void delete\_last(List) Удалить последний элемент
4. List copy(List) Создать копию заданного списка
5. void delete\_before(List, Node) Удаление элемента перед заданным
6. void delete\_after(List, Node) Удаление элемента после заданного
7. void delete(List, Node) Удаление заданного элемента
8. void insert\_before(List, Node, int) Вставка нового элемента перед заданным
9. void insert\_after(List, Node, int) Вставка нового элемента после заданного
10. void insert\_instead(List, Node, int) Вставка нового элемента вместо заданного
11. Node at(List, int index) Поиск элемента по индексу
12. Node at\_end(List, int index) Поиск элемента по индексу с конца (at\_end([1, 2, 3, 4], 1) -> 3)
13. Node find(List, int value) Поиск элемента по значению
14. Node find\_last(List, int value) Поиск элемента по значению с конца
15. int get\_length(List) Посчитать длину списка
16. void append(List, List) Добавление списка к началу другого (append([1, 2], [3]) -> [1, 2, 3])
17. List concat(List, List) Создать список, являющийся конкатенацией двух списков
18. void duplicate(List) Удвоить список ([1,2,3,4,5] -> [1,2,3,4,5,1,2,3,4,5])
19. List duplicate\_copy(List) Создать удвоенный список
20. List repeat(List, count) Создать у*count*енный список (repeat([1, 2], 3) -> [1, 2, 1, 2, 1, 2])
21. void swap\_pairs(List) Поменять соседние пары местами ([1,2,3,4,5] -> [2,1,4,3,5])
22. void reverse(List) Развернуть список ([1,2,3,4,5] -> [5,4,3,2,1])
23. List reversed(List) Создать развернутую копию списка.
24. void delete(List, int from, int to) Удалить элементы с индексами [from, to)
25. void keep(List, int from, int to) Удалить все, кроме элементов с индексами [from, to)
26. List sublist(List, int from, int to) Создать список из элементов с индексами [from, to)
27. List sublist\_copy(List, int from, int to) Создать список из элементов с индексами [from, to)
28. int get\_max(List) Вернуть значение максимального элемента; если список пустой, то ошибка
29. int get\_min(List) Вернуть значение минимального элемента; если список пустой, то ошибка
30. int sum(List) Вернуть сумму элементов списка.