**Лабораторная работа №8. Обобщённые коллекции**

1. **Описать интерфейс, включающий 7 методов. Создать класс, реализующие этот интерфейс для объектов типа string (выполнять с использованием класса LinkedList).**

**Написать программу, демонстрирующую работу с экземплярами класса.**

Номера заданий (v – номер варианта, n ‐ количество заданий == 30):

**Метод 1. (v + 7) % n + 1**

**Метод 2. (v + 14) % n + 1**

**Метод 3. (v + 21) % n + 1**

**Метод 4. (v + 28) % n + 1**

**Метод 5. (v + 35) % n + 1**

**Метод 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 список из списка List, повторяющегося count раз   
(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.

**Метод 7. (номер варианта = номер по списку%9)**

1. - в список L добавить новый элемент E1 перед каждым элементом E;   
   - из списка L удалить перед каждым вхождением элемента E один элемент, если он есть и отличен от E.
2. в непустой список L добавить пару новых элементов E1, E2 перед последним элементом, кратным 3;   
   - из списка L удалить пред- и предпредпоследний элементы, если они есть;
3. - в непустой список L добавить элементы которого упорядочены по неубыванию, новый элемент E так, чтобы сохранилась упорядоченность   
   - из упорядоченного списка L удалить все элементы большие E;
4. - в списке L удваивает каждое вхождение элемента E;   
   - из списка L удалить все элементы равные E
5. - в конец непустого списка L добавить копию его первого элемента;   
   - из списка L удалить второе вхождение первого элемента, если оно есть;
6. - в начало непустого списка добавить копию его последнего элемента;   
   - из списка L удалить все элементы равные последнему;
7. - в конец списка L1 добавить все элементы списка L2;   
   - из списка L1 удалить все элементы равные первому элементу списка L2;
8. - в начало списка L1 добавить все элементы списка L2;   
   - из списка L1 удалить все элементы равные последнему элементу списка L2;
9. - в начало списка L1 добавить первый элемент списка L2;   
   - из списка L1 удалить первый из элементов равный последнему элементу списка L2;
10. **Написать программу проверяющую знание английских слов. В текстовом файле хранятся английские слова и их перевод на русский язык.** Пользователю по очереди предлагается несколько английских слов из файла. Пользователь должен написать правильный перевод этих слов на русский язык. За правильный ответ начисляются баллы. Результаты проверки (имя пользователя, баллы, сколько слов было предложено, дата проверки) записываются в текстовый файл.