Во всех задачах предполагается наличие определения:

typedef struct Node {

int value;

struct Node \*next;

} Node;

1. Добавление элемента в начало списка

Реализуйте функцию, добавляющую элемент в начало списка. Поскольку указатель на первый элемент будет изменён, функция должна принимать указатель на указатель.

void push\_left(Node \*\* head, int value);

1. Удаление всех элементов списка

Реализуйте функцию очистки списка. С освобождением памяти и записью NULL в указатель на первый элемент:

void clear\_list(Node \*\* head);

1. Удаление элемента по заданному индексу (подумайте, почему функция должна принимать указатель на указатель) с возвратом его значения:

int pop(Node \*\* head, int index);

1. Удаление первого элемента списка с данным значением:

void remove(Node \*\* head, int value);

1. Удаление всех элементов списка с данным значением:

void remove\_all(Node \*\* head, int value);

1. Изменение всех элементов списка с данным значением на новое.

void replace\_all(Node \* head, int old\_value, int new);

1. Определение, сколько различных значений содержится в списке.

int unique(Node \* head);

1. Изменение порядка элементов на обратный.

void reverse(Node \*\* head);

9\*. Сортировка элементов списка двумя способами (изменение указателей, изменение значений элементов).