31. Линейный односвязный список.
План ответа:
	1. описание типа;
	2. основные операции.

1. описание типа;
	head->data1->data2->data3->NULL
	Отличия списка от массива
		Размер массива фиксирован, а списка нет.
		Списки можно переформировывать, изменяя несколько указателей.
		При удалении или вставки нового элемента в список адрес остальных не меняется.
		Позволяют при вставке нового элемента и удалении некоторого элемента не изменять адреса остальных элементов;
		Занимают больше места, так как каждый элемент списка должен содержать указатель на следующий элемент.

```
	struct list
	{
	    int value;
	    struct list *next;
	};

	struct list *create_node(int value)
	{
	    struct list *node = malloc(sizeof(struct list));
	    if (node)
	    {
	        node->value = value;
	        node->next = NULL;
	    }
	    return node;
	}
```

2. основные операции.
	NB: функции, изменяющие список, должны возвращать указатель на новый первый элемент.

	2.1 Добавление элемента в список

```
		struct list *add_front(struct list *head, struct list *node)
		{
		    node->next = head;
		    return node;
		}

		struct list *add_end(struct list *head, struct list *node)
		{
		    struct list *cur = head;
		    if (!head)
		        return node;

		    for (; cur->next; cur = cur->next);
		    cur->next = node;

		    return head;
		}
```

	2.2 Поиск

```
		struct list *search(struct list *head, int value)
		{
		    for ( ; head; head = head->next)
		        if (strcmp(head->value, value) == 0)
		            return head;

		    return NULL;
		}
```

	2.3 Обход всех (В данном случае печать)

```
		void print(struct list *head)
```

```
        {
            for (; head; head = head->next)
            {
                printf("%d ", head->value);
            }
            printf("\n");
        }
```

2.4 Удаление

```
        void free_all(struct list *head)
        {
            struct list *next;
            for (; head; head = next)
            {
                next = head->next;
                free(head);
            }
        }
```