1. Реализовать структуру для хранения представленных в задании данных. Информация в элемент структуры вводится со стандартного потока ввода.
2. Для хранения набора структур использовать указанный вид списка. Написать функцию вставки insert() в некоторое место списка, добавления в конец add(), удаления remove(), поиска и извлечения элемента find(), сохранения списка файл* save() и извлечения списка из файла* load(). Реализовать указанный вид поиска: под фильтром предполагаем удовлетворение условию >, <, ==, под поиском – только ==.
3. Фильтр возвращает новый список элементов, удовлетворяющих условию.

*кроме фильтра по указанным полям, можно написать универсальный фильтр по произвольному полю.

Вариант 1

Структура «Государство».
Минимальный набор полей: название, столица, язык, численность населения, площадь.
Список: односвязный стек.
Поиск по названию страны, фильтр по площади.

Вариант 2

Структура «Человек».
Минимальный набор полей: фамилия, имя, пол, рост, вес, дата рождения, телефон, адрес.
Список: двусвязная очередь.
Поиск по фамилии, фильтр по дате рождения.

Вариант 3

Структура «Школьник».
Минимальный набор полей: фамилия, имя, пол, класс, дата рождения, адрес.
Список: двусвязный упорядоченный (по фамилии) список.
Поиск по фамилии, фильтр по классу.

Вариант 4

Структура «Покупатель».
Минимальный набор полей: фамилия, имя, город, улица, номера дома и квартиры, номер счёта, средний сумма чека.
Список: двусвязный стек.
Поиск по фамилии, фильтр по средней сумме чека.

Вариант 5

Структура «Пациент».
Минимальный набор полей: фамилия, имя, дата рождения, телефон, адрес, номер карты, группа крови.
Список: односвязный стек.
Поиск по фамилии, фильтр по группе крови.

Вариант 6

Структура «Команда».
Минимальный набор полей: название, город, число побед, поражений, ничьих, количество очков.
Список: двусвязная очередь.
Поиск по названию, фильтр по числу побед.

Вариант 7

Структура «Стадион».
Минимальный набор полей: название, виды спорта, год постройки, вместимость, количество арен.
Список: двусвязный упорядоченный (по названию) список.
Поиск по названию, фильтр по году постройки.

Вариант 8

Структура «Автовладелец».
Минимальный набор полей: фамилия, имя, номер автомобиля, дата рождения, номер техпаспорта.
Список: двусвязный стек.
Поиск по номеру техпаспорта, фильтр по фамилии.

Вариант 9

Структура «Автомобиль».
Минимальный набор полей: марка, цвет, серийный номер, количество дверей, год выпуска, цена.
Список: односвязный стек.
Поиск по серийному номеру, фильтр по цене.

Вариант 10

Структура «Фильм».
Минимальный набор полей: фамилия, имя режиссёра, название, страна, год выпуска, стоимость, доход.
Список: двусвязный упорядоченный (по названию) список.
Поиск по названию, фильтр по доходу.

Вариант 11

Структура «Альбом».
Минимальный набор полей: имя или псевдоним исполнителя, название альбома, количество композиций, год выпуска.
Список: двусвязная очередь.
Поиск по названию альбома, фильтр по имени.

Вариант 12

Структуры «Комплексное число», «Уравнение с комплексными коэффициентами».
Минимальный набор полей: коэффициенты в уравнении , набор корней уравнения.
Список (из уравнений): двусвязный стек.
Поиск по коэффициентам.
Дополнительная функция: поиск корней уравнения.

Вариант 13

Структуры «Комплексное число», «Дробь».
Минимальный набор полей: комплексные коэффициенты a, b, c, d в выражении , набор корней уравнения.
Список (из дробно-линейных коэффициентов): односвязный стек.
Поиск по коэффициентам.
Дополнительная функция: поиск значения выражения.

Вариант 14

Структура «Матрица».
Минимальный набор полей: размерности, коэффициенты матрицы.

Список: двусвязная очередь.
Поиск по следу, фильтр по размерности.
Дополнительная функция: вычисление определителя и следа, произведения.

Вариант 15

Структура «Вектор».
Минимальный набор полей: размерность, коэффициенты вектора.
Список: двусвязный стек.
Поиск по длине, фильтр по размерности.
Дополнительная функция: вычисление длины вектора, скалярного произведения двух векторов.

Вариант 16

Структура «Определённый интеграл».
Минимальный набор полей: указатель на подынтегральную функцию, шаг вычисления (точность), пределы интегрирования.
Список: двусвязный кольцевой список.
Фильтр по значению интеграла.
Дополнительная функция: приближённое вычисление значения интеграла.

Вариант 17

Структура «Уравнение».
Минимальный набор полей: указатель на функцию f(x) в уравнении, шаг вычисления (точность), левая и правая границы области, в которой производится поиск решения уравнения.
Список: двусвязный стек.
Поиск по решениям уравнения.
Дополнительная функция: приближённое вычисление решения уравнения.

Вариант 18

Структура «Государство».
Минимальный набор полей: название, столица, язык, численность населения, площадь.
Список: двусвязная очередь.
Поиск по численности населения, фильтр по языку.

Вариант 19

Структура «Человек».
Минимальный набор полей: фамилия, имя, пол, рост, вес, дата рождения, телефон, адрес.
Список: односвязный стек.
Поиск по фамилии, фильтр по адресу.

Вариант 20

Структура «Школьник».
Минимальный набор полей: фамилия, имя, пол, класс, дата рождения, адрес.
Список: двусвязный стек.
Поиск по фамилии, фильтр по дате рождения.

Вариант 21

Структура «Покупатель».
Минимальный набор полей: фамилия, имя, город, улица, номера дома и квартиры, номер счёта.
Список: двусвязный упорядоченный (по фамилии) список.

Поиск по номеру счета, фильтр по городу.

Вариант 22

Структура «Пациент».
Минимальный набор полей: фамилия, имя, дата рождения, телефон, адрес, номер карты, группа крови.
Список: двусвязная очередь.
Поиск номеру карты, фильтр по группе крови..

Вариант 23

Структура «Команда».
Минимальный набор полей: название, город, число побед, поражений, ничьих, количество очков.
Список: односвязный стек.
Поиск по названию, фильтр по числу побед.

Вариант 24

Структура «Автовладелец».
Минимальный набор полей: фамилия, имя, номер автомобиля, дата рождения, номер техпаспорта.
Список: двусвязный упорядоченный (по фамилии) список.
Поиск по номеру автомобиля, фильтр по имени.

Вариант 25

Структура «Государство».
Минимальный набор полей: название, столица, язык, численность населения, площадь.
Список: односвязный стек.
Поиск по столице, фильтр по численности населения.

Вариант 26

Структура «Программист».
Минимальный набор полей: фамилия, имя, email, skype, telegram, основной язык программирования, текущее место работы, уровень(число от 1 до 10).
Список: кольцевой односвязный список.
Поиск по email, фильтр по уровню.

Вариант 27

Структура «Спортсмен».
Минимальный набор полей: фамилия, имя, возраст, гражданство, вид спорта, количество медалей.
Список: двусвязная очередь.
Поиск по фамилии, фильтр по числу медалей.

Вариант 28

Структура «Полином».
Минимальный набор полей: степень полинома, коэффициенты
Список: двусвязный упорядоченный (по степени) список.
Поиск по коэффициентам, фильтр по степени полинома.
Дополнительная функция: поиск корня полинома на указанном отрезке.

Вариант 29

Структура «Профиль в соц.сети».

Минимальный набор полей: псевдоним, адрес страницы, возраст, количество друзей, интересы, любимая цитата.
Список: двусвязный стек.
Поиск по псевдониму, фильтр по количеству друзей.

Вариант 30

Структура «Велосипед».
Минимальный набор полей: марка, тип, тип тормозов, количество колес, диаметр колеса, наличие амортизаторов, детский или взрослый.
Список: односвязная очередь.
Поиск по марке, фильтр по диаметру колеса.

Вариант 31

Структура «Ноутбук».
Минимальный набор полей: производитель, модель, размер экрана, процессор, количество ядер, объем оперативной памяти, объем диска, тип диска, цена.
Список: односвязный стек.
Поиск по марке, фильтр по цене.

Вариант 32

Структура «Смартфон».
Минимальный набор полей: марка, размер экрана, количество камер, объем аккумулятора, максимальное количество часов без подзарядки, цена.
Список: односвязный упорядоченный (по стоимости) список.
Поиск по марке, фильтр по размеру экрана.

Вариант 33

Структура «Фотоаппарат».
Минимальный набор полей: производитель, модель, тип, размер матрицы, количество мегапикселей, вес, тип карты памяти, цена.
Список: двусвязный стек.
Поиск по модели, фильтр по размеру матрицы.

Вариант 34

Структура «Супергерой».
Минимальный набор полей: псевдоним, настоящее имя, дата рождения, пол, суперсила, слабости, количество побед, рейтинг силы.
Список: двусвязная очередь.
Поиск по псевдониму, фильтр по рейтингу силы.

Вариант 35

Структура «Сериал».
Минимальный набор полей: название, режиссер, количество сезонов, популярность, рейтинг, дата запуска, страна.
Список: односвязный стек.
Поиск по режиссеру, фильтр по популярности.

Вариант 36

Структура «Программа».
Минимальный набор полей: название, версия, лицензия, есть ли версия для android, iOS, платная ли, стоимость, разработчик, открытость кода, язык кода.
Список: односвязный упорядоченный (по стоимости) список.

Поиск по названию, фильтр по платности.

Пример кода:

```
#include <stdlib.h>
#include <stdio.h>

struct Element
{
	int Key;
	struct Element* next;
	//Element* prev;
};

typedef struct Element Element;

void InitElement(Element* e, int K)
{
	e->next = NULL;
	e->Key = K;
}

//для хранения односвязного списка достаточно иметь указатель только на его начало
typedef struct
{
	Element* First;
} List;


void InitList(List* L)
{
	L->First = NULL;
}

//вставить в конец списка L элемент e
void InsertEnd(List* L, Element* e)
{
	if(L->First==NULL)
	{
		//пустой список
		L->First = e;
		return;
	}
	//ищем последний элемент - это можно сделать рекурсивно
	Element* Current = L->First;
	while(Current->next!=NULL)
	{
		Current = Current->next;
	}

	//нашли конец списка - место для вставки
```

```
        e->next = NULL;
        Current->next = e;
}

//поиск элемента e в списке L
Element* Search(List* L, Element* e)
{
        Element* Current = L->First;
        //здесь можно было бы ввести указатель на функцию сравнения - на
случай, если список состоит из Ваших собственных структур
        while(Current!=NULL && Current->Key!=e->Key)
        {
                Current = Current->next;
        }
        //вышли из цикла, если закончился список или нашли нужный элемент
        return Current;
}

//вставить в список L элемент e после элемента WhereToInsert
void Insert(List* L, Element* WhereToInsert, Element* e)
{
        if(L->First==NULL)
        {
                //пустой список
                L->First = e;
                return;
        }

        Element* p = Search(L, WhereToInsert);
        if(p==NULL) return;
        e->next = WhereToInsert->next;
        WhereToInsert->next = e;
}

//определить следующий элемент за e
Element* Successor(List* L, Element* e)
{
        Element* p = Search(L, e);
        if(p!=NULL) return p->next;
        return NULL;
}

//определить предыдущий элемент перед e
Element* Predecessor(List* L, Element* e)
{
        Element* Current = L->First;
        if(Current==NULL) return NULL;
        if(Current->Key==e->Key) return NULL;

        //здесь можно было бы ввести указатель на функцию сравнения - на
случай, если список состоит из Ваших собственных структур
        while(Current->next!=NULL && Current->next->Key!=e->Key)
        {
                Current = Current->next;
```

```
        }
        //вышли из цикла, если закончился список или нашли нужный элемент
        return Current;
}

void Free(List*L, Element* e)
{
        //освободить ресурсы, выделенные под 1 элемент
        free(e);
}

//удалить элемент e
void Delete(List* L, Element* e)
{
        Element* p = Search(L, e);
        if(p==NULL) return;

        Element* prev = Predecessor(L, e);
        Element* next = Successor(L, e);

        prev->next = next;
        Free(L, p);
}

void MakeAction(List* L, void (*f_ptr)(Element*), Element* e)
{
        if(e==NULL)
                e=L->First;
        if(e==NULL) return;

        f_ptr(e);

        if(e->next!=NULL)
        MakeAction(L, f_ptr, e->next);
}

void print_elem(Element* e)
{
        if(e==NULL)
                printf("\nNULL");
        printf("\nKey = %d\tNext = %p", e->Key, e->next);
}

int main()
{
        Element* a = (Element*)malloc(sizeof(Element)); InitElement(a, 10);
        Element* b = (Element*)malloc(sizeof(Element)); InitElement(b, 30);
        Element* c = (Element*)malloc(sizeof(Element)); InitElement(c, 20);

        List L; InitList(&L);
        InsertEnd(&L, a);InsertEnd(&L, b);Insert(&L, b, c);

        Element x; x.Key = 30;
        Element* p = Search(&L, &x);
```

```
        if(p==NULL) printf("\nNot found");
        else printf("\nSearch: Key = %d", p->Key);

        x.Key = 20;
        p = Predecessor(&L, &x);
        if(p==NULL) printf("\nNot found");
        else printf("\nPredecessor: Key = %d", p->Key);

        x.Key = 10;
        p = Successor(&L, &x);
        if(p==NULL) printf("\nNot found");
        else printf("\nSuccessor: Key = %d", p->Key);


        void (*p_print)(Element*); p_print = print_elem;
        MakeAction(&L, p_print, NULL);
        Delete(&L, b);
                printf("\nDeletion\n-----------\n");
        MakeAction(&L, p_print, NULL);

        return 0;
}
```