##### Задание 5

1. Односвязный список имеет следующую структуру узла

typedef struct node {

int value;

struct node \*next;

} node;

Напишите функцию add(корень, значение) для добавления значения в начало списка.

#include <stdio.h>

#include <stdlib.h>

typedef struct node {

int value;

struct node \*next;

} node;

// определение функции add

...

int main()

{ node \*root=NULL;

// root->NULL

add(&root,1);

// root->[1]->NULL

add(&root,2);

// root->[2]->[1]->NULL

}

2. Односвязный список имеет следующую структуру узла

typedef struct node {

int value;

struct node \*next;

} node;

Напишите функцию print(корень) для печати содержимого списка.

#include <stdio.h>

#include <stdlib.h>

typedef struct node {

int value;

struct node \*next;

} node;

// определение функции print

...

int main()

{ node \*root=NULL;

// root->NULL

add(&root,1);

// root->[1]->NULL

add(&root,2);

// root->[2]->[1]->NULL

print(root);

// 2 1

}

3. Односвязный список имеет следующую структуру узла

typedef struct node {

int value;

struct node \*next;

} node;

Напишите функцию find(корень, значение), которая возвращает указатель на узел, имеющий заданное значение. Если такого значения нет, то функция должна возвращать NULL.

#include <stdio.h>

#include <stdlib.h>

typedef struct node {

int value;

struct node \*next;

} node;

// определение функции find

...

int main()

{ node \*root=NULL;

// root->NULL

add(&root,1);

// root->[1]->NULL

add(&root,2);

// root->[2]->[1]->NULL

node \*n1=find(root,1);

if(n1) n1->value=3; // заменим 1 на 3

n1=find(root,1);

if(n1==NULL) printf("NULL\n"); // NULL

}

4. Односвязный список имеет следующую структуру узла

typedef struct node {

int value;

struct node \*next;

} node;

Напишите функцию delall(корень, значение), которая удаляет все узлы со значением равным заданному.

#include <stdio.h>

#include <stdlib.h>

typedef struct node {

int value;

struct node \*next;

} node;

// определение функции delall

...

int main()

{ node \*root=NULL;

// root->NULL

add(&root,1);

// root->[1]->NULL

add(&root,2);

// root->[2]->[1]->NULL

add(&root,1);

// root->[1]->[2]->[1]->NULL

delall(&root,1);

// root->[2]->NULL

}

5. Односвязный список имеет следующую структуру узла

typedef struct node {

int value;

struct node \*next;

} node;

Напишите функцию insertafter(узел, значение), которая вставляет узел с заданным значением после указанного узла.

#include <stdio.h>

#include <stdlib.h>

typedef struct node {

int value;

struct node \*next;

} node;

// определение функции insertafter

...

int main()

{ node \*root=NULL;

// root->NULL

add(&root,1);

// root->[1]->NULL

add(&root,2);

// root->[2]->[1]->NULL

add(&root,3);

// root->[3]->[2]->[1]->NULL

node \*n=find(root,2);

insertafter(n,5);

// root->[3]->[2]->[5]->[1]->NULL

}

6. Двусвязный список имеет следующую структуру узла

typedef struct node {

int value;

struct node \*next, \*prev;

} node;

Напишите функцию delnode(корень, узел), которая удаляет указанный узел. Если узел является первым узлом списка (корнем), то изменить указатель на корень.

#include <stdio.h>

#include <stdlib.h>

typedef struct node {

int value;

struct node \*next, \*prev;

} node;

// определение функции delnode

...

int main()

{ node \*root=NULL;

// root->NULL

add(&root,1);

// root->[1]->NULL

add(&root,2);

// root->[2]<->[1]->NULL

add(&root,3);

// root->[3]<->[2]<->[1]->NULL

node \*n=root->next;

delnode(&root,n);

// root->[3]<->[1]->NULL

}

7. Двусвязный список имеет следующую структуру узла

typedef struct node {

int value;

struct node \*next, \*prev;

} node;

Напишите функцию insertafter(корень, узел, значение), которая добавляет новый узел с указанным значением после заданного. Для вставки нового узла в начало списка в качестве узла указывается NULL, при этом изменяется указатель на корень.

#include <stdio.h>

#include <stdlib.h>

typedef struct node {

int value;

struct node \*next, \*prev;

} node;

// определение функции insertafter

...

int main()

{ node \*root=NULL;

// root->NULL

insertafter(&root,NULL,1);

// root->[1]->NULL

insertafter(&root,NULL,2);

// root->[2]<->[1]->NULL

insertafter(&root,NULL,3);

// root->[3]<->[2]<->[1]->NULL

node \*n=root->next;

insertafter(&root,n,4);

// root->[3]<->[2]<->[4]<->[1]->NULL

}

8. Двусвязный список имеет следующую структуру узла

typedef struct node {

int value;

struct node \*next, \*prev;

} node;

На основе такого списка создается структура данных "дек" (двойная очередь, в которую можно добавлять и убирать элементы с обоих сторон).

typedef struct deque {

int size;

node \*front, \*back;

} deque;

Для работы с такой очередью нужно использовать функции:

void deque\_init(deque\* d) { d->size=0; d->front=d->back=NULL; } // инициализация

int deque\_front(deque\* d) { return d->front->value; } // первый элемент

int deque\_back(deque\* d) { return d->back->value; } // последний элемент

int deque\_size(deque\* d) { return d->size; } // размер

void deque\_pushfront(deque\* d, int val) // добавить значение в начало

{ node \*n=malloc(sizeof(node)); // новый узел

n->value=val;

n->next=d->front;

n->prev=NULL;

d->size++;

if(d->front==NULL) {

d->front=d->back=n; // оба на новый узел

}

else {

d->front->prev=n;

d->front=n;

}

}

void deque\_popfront(deque\* d) // удалить значение в начале

{

node \*n=d->front;

d->front=n->next;

free(n);

d->size--;

if(d->size==0)

d->front=d->back=NULL;

if(d->front)

d->front->prev=NULL;

}

void deque\_pushback(deque\* d, int val); // добавить значение в конец

void deque\_popback(deque\* d); // удалить значение в конце

// эти 2 функции вы должны написать самостоятельно

Пример использования

#include <stdio.h>

#include <stdlib.h>

// определение node, deque и функций

...

int main()

{

deque d;

deque\_init(&d);

deque\_pushfront(&d,1);

deque\_pushfront(&d,2);

deque\_pushback(&d,3);

// 2 1 3

printf("%d\n",deque\_front(&d)); // 2

deque\_popback(&d);

printf("%d\n",deque\_back(&d)); // 1

}

9. Оби-Ван Кеноби – один из последних представителей ордена рыцарей джедаев. Оби-Ван Кеноби родом с планеты Стьюджон, где абсолютно все жители чтут порядок, и Оби-Ван Кеноби – не исключение.

Во всех эпизодах "Звездных войн" Оби-Ван Кеноби орудует синим световым мечом, но мало кто знает, что на самом деле у него их много, причем все мечи Кеноби пронумерованы. Оби-Ван Кеноби хранит все свои мечи на очень длинном столе. Когда он хочет вооружиться, то он берет самый правый меч со стола, и идет по своим делам. Оби-Ван Кеноби орудует взятым мечом пока не потеряет или не сломает его.

Иногда кто-нибудь дарит Оби-Вану Кеноби новый меч, и тогда он просто кладет его на стол справа к тем, что уже лежат там. Но самое страшное случается тогда, когда к столу подходит мама Оби-Вана Кеноби. Мама тоже житель планеты Стьюджон, и поэтому она всегда возмущается, что мечи лежат неупорядоченно. Чтобы исправить это, она забирает всю левую половину мечей (если мечей на столе было нечетное количество, то мама забирает наибольшее целое число мечей, меньшее половины их общего количества) и начинает их подкладывать справа к оставшимся. Причем сначала она кладет меч, который изначально был самым левым, потом правее от него кладет второй меч и так далее.

Определите, в каком порядке лежат мечи у Оби-Вану Кеноби на данный момент. У каждого меча есть свой личный номер, который известен только Оби-Вану Кеноби. Ну и, конечно, вам.

В первой строке входного файла дано натуральное числа *n*  – количество изменений, произошедших на столе (1 ≤ *n* ≤ 106). В следующих *n* строках заданы описания изменений в следующем формате:

* если строка начинается со слова add, то в той же строке через пробел находится число *x* (1 ≤ *x* ≤ *n*) – личный номер меча, подаренного Оби-Вану Кеноби. Гарантируется, что мечей с таким номером Оби-Вану Кеноби раньше не дарили и больше не подарят.
* если строка содержит единственное слово take, то это обозначает, что Оби-Ван Кеноби забрал самый правый меч со стола с собой. Если на столе не было мечей, то ничего не произошло.
* если же строка содержит единственное слово mum!, то это обозначает, что к столу подошла мама и навела порядок. Если на столе было меньше двух мечей, то ничего не произошло.

Изначально стол был пуст.

В первой строке выходного файла выведите число *k*  – количество мечей, лежащих на данный момент на столе у Оби-Вана Кеноби. В следующей строке выведите *k* чисел через пробел – личные номера мечей, лежащих на столе, в том порядке, в котором они на нем лежат (слева направо). Если *k*=0, то вторую строку не выводить.

10. Дисковое пространство поделено на так называемые кластеры. При записи на диск файлы разделяются на части размером с кластер. Если все части файла размещены последовательно в последовательных кластерах диска, то последовательное чтение файла происходит максимально быстро. При изменении файла отдельные его части (фрагменты) могут оказаться разбросанными по всему диску в произвольном порядке, т. е. файл становится фрагментированным. При этом быстродействие компьютера снижается. Обычно дефраментация проводится, когда число фрагментированных файлов составляет 10-20% или более от числа всех файлов. Для определения степени фрагментации можно использовать таблицу размещения файлов FAT (File Allocation Table). Упрощенно ее можно описать следующим образом. Дисковое пространство делится на *M* кластеров, нумерующихся с 1 до *M*. В таблице FAT для каждого кластера диска записывается число 0, если соответствующий кластер свободен, или число −1, если это последний кластер файла, или число от 1 до *M*, указывающее на следующий кластер, занятый файлом.

Во входном файле в первой строке содержится количество кластеров на диске *M* (1 ≤ *M* ≤ 10000), в последующих *M* строках – таблица FAT. Данные корректны, т.е. не содержат циклов и пересечений.

В выходной файл вывести в первой строке вывести общее число файлов на диске, во второй строке – число фрагментированных файлов. Файл будем считать не фрагментированным, если для всех частей файла верно, что *i* часть файла занимает *b*+*i*−1 кластер, где *b* – первый кластер файла.