#include <stdio.h>

#include <cs50.h>

#include <string.h>

#include <ctype.h>

//создали структуру sllnode включающую в себя

// val типа int

// и указатель next

typedef struct sllist

{

int val;

struct sllist\* next;

}

sllnode;

//прототипы функций

sllnode\* create(int value);

bool find(sllnode\* head, int value);

sllnode\* insert (sllnode\* head, int value);

void destroy(sllnode\* head);

int count(sllnode\* head);

void showlist (sllnode\* head);

sllnode\* delelement (sllnode\* head, int value);

void showelement (sllnode\* head, int number);

int main(void)

{

//создали singlelist и присвоили val значение 6.

sllnode\* new = create(6);

new = insert(new, 10);

new = insert(new, 5);

new = insert(new, 4);

new = insert(new, 3);

showlist(new);

new = delelement(new, 0);

showlist(new);

new = delelement(new, 2);

showlist(new);

showelement(new, 1);

printf("%i\n", count(new));

printf("%i\n", new->val);

destroy(new);

}

// выводит элемент под номером number в списке (нумерация с 0, как в массивах)

void showelement (sllnode\* head, int number)

{

sllnode\* ptr = head;

for (int i = 0; i<number; i++)

{

ptr = ptr->next;

}

printf("%i\n", ptr->val);

}

// удаляет элемент под номером number

sllnode\* delelement (sllnode\* head, int number)

{

// проверка на 1 элемент массива

if (number == 0)

{

sllnode\* ptr = head;

ptr = head->next;

free(head);

return ptr;

}

else

{

sllnode\* ptrcurr = head;

sllnode\* ptrprev = head;

for (int i = 0; i < number; i ++)

{

ptrcurr = ptrcurr->next;

}

for (int i = 0; i < number-1; i ++)

{

ptrprev = ptrprev->next;

}

ptrprev->next = ptrcurr->next;

free(ptrcurr);

return head;

}

}

// выводит весь связанный список

void showlist (sllnode\* head)

{

if (head != NULL)

{

printf("%i ", head->val);

showlist(head->next);

}

else

{

printf("\n");

}

}

// подсчёт числа элементов в списке

int count(sllnode\* head)

{

int counter = 0;

while (head->next != NULL)

{

head = head->next;

count(head);

counter++;

}

counter++;

return counter;

}

//удаляем связанный список

void destroy(sllnode\* head)

{

if (head->next != NULL)

{

destroy(head->next);

head->next = NULL;

}

if (head->next == NULL)

{

free(head);

}

}

//добавлеяем ещё один элемент типа sllnode

sllnode\* insert (sllnode\* head, int value)

{

//выделили память под переменную типа sllnode

sllnode\* ptr = malloc(sizeof(sllnode));

//проверили на ошибки

if (ptr == NULL)

{

return NULL;

}

//заполнили новый элемент данными

ptr->val = value;

ptr->next = head;

//вернули указатель на первый элемент списка

return ptr;

}

//функция ищет в sllnode значение value

bool find(sllnode\* head, int value)

{

sllnode\* ptr = head;

if (ptr->val == value)

{

return true;

}

while (ptr->next != NULL)

{

if (ptr->val == value)

{

return true;

}

else

{

ptr = ptr->next;

}

}

return false;

}

//функция создаёт перменную типа sllnode

sllnode\* create(int value)

{

//выделили память под переменную типа sllnode

sllnode\* ptr = malloc(sizeof(sllnode));

//проверили на ошибки

if (ptr == NULL)

{

return NULL;

}

ptr->val = value;

ptr->next = NULL;

return ptr;

}