Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №2.2

з дисципліни
«Алгоритми і структури даних»

Виконав:
студент групи ІМ-21
Сірик Максим Олександрович
номер у списку групи: 19

Перевірила:
Молчанова А. А.

Київ 2021

Завдання

1. Створити список з n (n>0) елементів (n вводиться з клавіатури), якщо інша кількість елементів не вказана у конкретному завданні.
2. Тип ключів (інформаційних полів) задано за варіантом.
3. Значення елементів списку взяти самостійно такими, щоб можна було продемонструвати коректність роботи алгоритму програми. Введення значень елементів списку можна виконати довільним способом (випадкові числа, формування значень за формулою, введення з файлу чи з клавіатури).
4. Вид списку (черга, стек, дек, прямий однозв'язний лінійний список, обернений однозв'язний лінійний список, двозв'язний лінійний список, однозв'язний кільцевий список, двозв'язний кільцевий список) вибрати самостійно з метою найбільш доцільного рішення поставленої за варіантом задачі.
5. Виконати над створеним списком дії, вказані за варіантом, та коректне звільнення пам'яті списку.
6. При виконанні заданих дій, виводі значень елементів та звільненні пам'яті списку вважати, що довжина списку (кількість елементів n чи 2n) невідома на момент виконання цих дій.
7. Повторювані частини алгоритму необхідно оформити у вигляді процедур або функцій (для створення, обробки, виведення та звільнення пам'яті списків) з передачею списку за допомогою параметра(ів).

Варіант 19

Задано два списки, список S1 довжиною 2n елементів і список S2 довжиною n елементів. Ключами елементів обох списків є натуральні числа. Вставити список S2 у середину списку S1, не використовуючи додаткових структур даних, крім простих змінних (тобто «на тому ж місці»).

Текст програми

"main.c"

```
#include "List.c"
```

```
int main() {
  int n = 0;
  scanf("%d", &n);

  List *s1 = init_list();
  List *s2 = init_list();

  for (int i = 0; i < n * 2; i++)
        add_value(s1, (i + 1) * 10);

  for (int i = 0; i < n; i++)
        add_value(s2, i + 1);

  print_list(s1);
  print_list(s2);

  printf("\nAfter combining\n\n");

  List *list = combine_lists_center(&s1, &s2);

  // s1 and s2 now null to avoid memory leaks
  print_list(s1);
  print_list(s2);
  print_list(list);

 free_list(list);
 return 0;
}
```

“List.c”

```
#include "./utils/print_color.c"
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
  unsigned int value;
  struct Node *next;
} Node;

typedef struct List {
  Node *head;
  Node *tail;
  int size;
} List;

List *init_list() {
  List *list = malloc(sizeof(List));
  list->head = NULL;
  list->tail = NULL;
  list->size = 0;
  return list;
}

void add_value(List *list, unsigned int value) {
  Node *node = (Node *)malloc(sizeof(Node));
  node->value = value;
  node->next = NULL;
```

```
  if (list->head == NULL) {
        list->head = node;
        list->tail = list->head;
  } else {
        list->tail->next = node;
        list->tail = list->tail->next;
  }

  list->size++;
}

void print_list(List *list) {
  set_info_color();

  if (list == NULL) {
        printf("List is NULL\n");
  } else if (list->head == NULL) {
        printf("List is empty\n");
  } else {
        Node *current = list->head;

        reset_color();
        printf("\nHead: %d\n", list->head->value);
        printf("Tail: %d\n", list->tail->value);
        set_info_color();

        printf("%d: ", list->size);

        while (current != NULL) {
```

```
        printf("%d-> ", current->value);
        current = current->next;
        }

        printf("NULL\n");
  }

  reset_color();
}

void free_list(List *list) {
  Node *current = list->head;
  Node *next = NULL;

  set_urgent_color();
  printf("Freeing: ");

  while (current != NULL) {
        printf("%d ", current->value);
        next = current->next;
        free(current);
        current = next;
  }

  reset_color();
  printf("\n");
}

// Combines two lists into one list, where the second list is inserted in the
// middle of the first list; both pointers would be null, return pointer to a
```

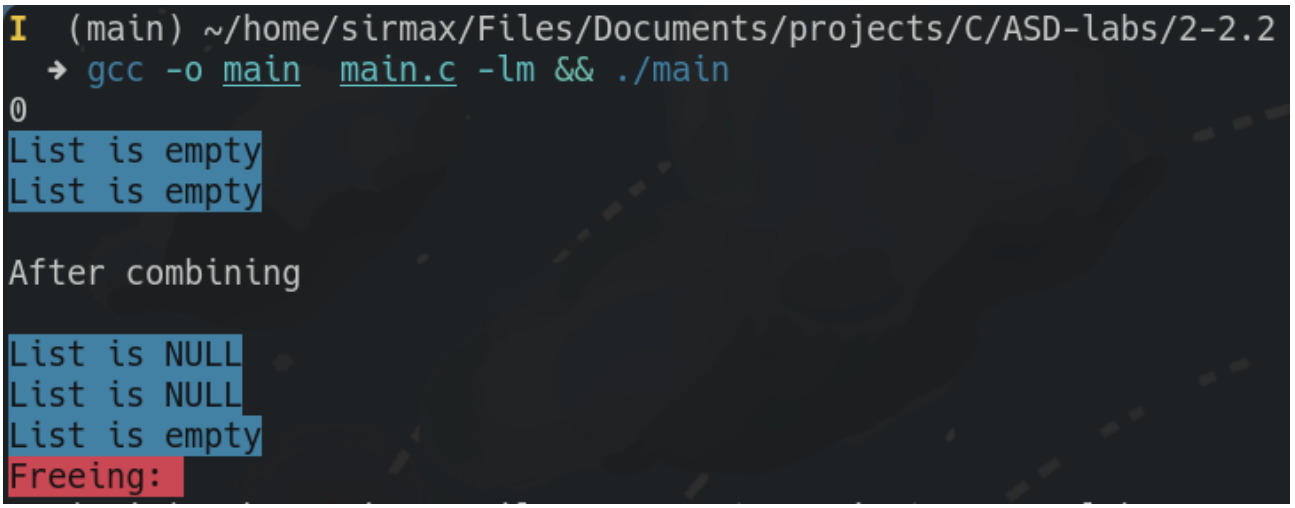
```
// new list;
List *combine_lists_center(List **list1ptr, List **list2ptr) {
  List *list1 = *list1ptr;
  List *list2 = *list2ptr;

  Node *slow = list1->head;
  Node *fast = list1->head;

  while (fast != NULL && fast->next != NULL) {
        fast = fast->next->next;

        if (fast != NULL)
        slow = slow->next;
  }

  // if the first list is empty
  if (slow == NULL) {
        List *new_list = list2;
        free(list1);
        *list2ptr = NULL;
        *list1ptr = NULL;

        return new_list;
  }

  // slow will be in the middle of the list
  Node *tmp = slow->next;
  slow->next = list2->head;
  list2->tail->next = tmp;
```

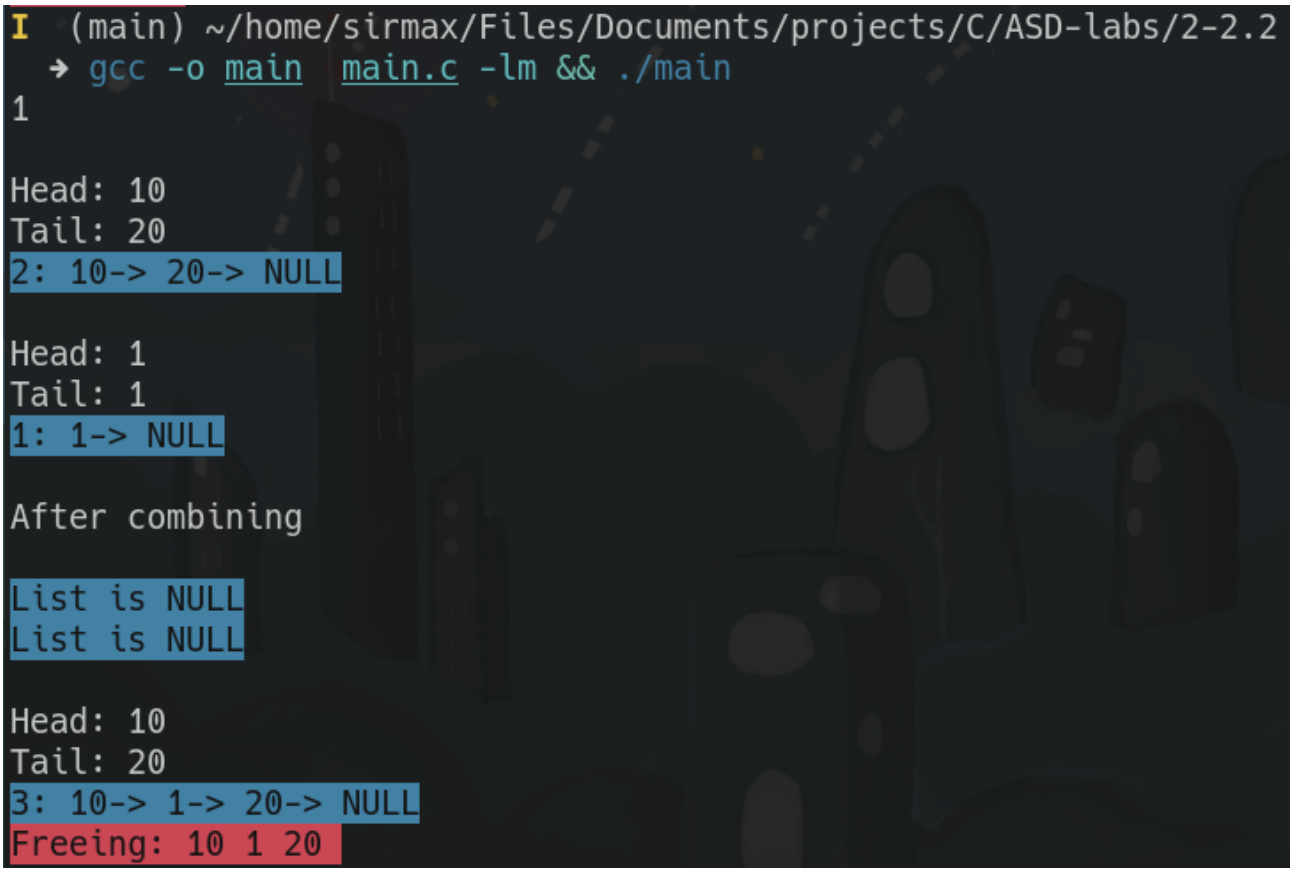
```
  list1->size += list2->size;

  List *new_list = list1;
  free(list2);
  *list2ptr = NULL;
  *list1ptr = NULL;

  return new_list;
}
```

“print_colors.c”

```
#include <stdio.h>

void set_urgent_color() {
  printf("\033[41;30m");
}

void set_info_color() {
  printf("\033[44;30m");
}

void reset_color() {
  printf("\033[0m");
}
```

Результати тестування




I (main) ~/home/sirmax/Files/Documents/projects/C/ASD-labs/2-2.2
➜ gcc -o main main.c -lm && ./main
1
Head: 10
Tail: 20
2: 10-> 20-> NULL
Head: 1
Tail: 1
1: 1-> NULL
After combining
List is NULL
List is NULL
Head: 10
Tail: 20
3: 10-> 1-> 20-> NULL
Freeing: 10 1 20