Operating Systems (EE463)

Lab4: The C Programming Environment in Linux

Name	ID
Abdullah Essam Alghamdi	1939627

Instructor: Eng. Turkey Saderaldin

Ex1:

```
// main.c
// This file contains a sample program used in the
// gdb debugging tutorial.
#include <stdio.h>
#include <stdlib.h>

int num_instantiated = 0;

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

Node *create_node(const int value, Node *next) {
    Node *new_node = (Node *)malloc(sizeof(Node));
    new_node->value = value;
    new_node->next = next;
    printf("Creating Node, %d are in existence right now\n", ++num_instantiated);
    return new_node;
}

void destroy_node(Node *node) {
    if (node) {
        printf("Destroying Node, %d are in existence right now\n", --num_instantiated);
        free(node);
    }
}
```

```
typedef struct LinkedList {
    Node *head;
} LinkedList;

LinkedList *create_linkedlist() {
    LinkedList *list = (LinkedList *)malloc(sizeof(LinkedList));
    list->head = NULL;
    return list;
}

void destroy_linkedlist(LinkedList *list) {
    Node *marker = list->head;
    while (marker != NULL) {
        Node *temp = marker->next;
        destroy_node(marker);
        marker = temp;
    }
    free(list);
}

// returns 0 on success, -1 on failure
int insert(LinkedList *list, const int new_element) {
    Node *new_node = create_node(new_element, list->head);
    list->head = new_node;
    return (list->head != NULL) ? 0 : -1;
}
```

```
// returns 0 on success, -1 on failure
int remove_item(LinkedList *list, const int element_to_remove) {
    Node *marker = list->head;
    Node *temp = NULL;

    while (marker != NULL) {
        if (marker->value == element_to_remove) {
            if (temp == NULL) {
                if (marker->next == NULL) {
                    list->head = NULL;
                    destroy_node(marker);
                    marker = NULL;
                } else {
                    list->head = create_node(marker->value, marker->next);
                    destroy_node(marker);
                    marker = NULL;
                }
                return 0;
            } else {
                temp->next = marker->next;
                destroy_node(marker);
                temp = NULL;
                return 0;
            }
        }
        temp = marker;
        marker = marker->next;
    }
    return -1; // failure
}

void print_list(const LinkedList *list) {
    Node *marker = list->head;
    while (marker != NULL) {
        printf("%d\n", marker->value);
        marker = marker->next;
    }
}

int main(int argc, char **argv) {
    LinkedList *list = create_linkedlist();

    insert(list, 1);
    insert(list, 2);
    insert(list, 3);
    insert(list, 4);

    printf("The fully created list is:\n");
    print_list(list);

    printf("\nNow removing elements:\n");
    remove_item(list, 4);
    print_list(list);
    printf("\n");

    remove_item(list, 1);
    print_list(list);
    printf("\n");

    remove_item(list, 2);
    print_list(list);
    printf("\n");

    remove_item(list, 3);
    print_list(list);

    destroy_linkedlist(list);

    return 0;
}
```

Ex2:

```
// main.c
// This file contains a sample program used in the
// gdb debugging tutorial.
#include <stdio.h>
#include <stdlib.h>

int num_instantiated = 0;

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

Node *create_node(const int value, Node *next) {
    Node *new_node = (Node *)malloc(sizeof(Node));
    new_node->value = value;
    new_node->next = next;
    printf("Creating Node, %d are in existence right now\n", ++num_instantiated);
    return new_node;
```

```
}

void destroy_node(Node *node) {
    if (node) {
        printf("Destroying Node, %d are in existence right now\n", --
num_instantiated);
        free(node);
    }
}

typedef struct LinkedList {
    Node *head;
} LinkedList;

LinkedList *create_linkedlist() {
    LinkedList *list = (LinkedList *)malloc(sizeof(LinkedList));
    list->head = NULL;
    return list;
}

void destroy_linkedlist(LinkedList *list) {
    Node *marker = list->head;
    while (marker != NULL) {
        Node *temp = marker->next;
        destroy_node(marker);
        marker = temp;
    }
    free(list);
}

// returns 0 on success, -1 on failure
int insert(LinkedList *list, const int new_item) {
    Node *new_node = create_node(new_item, list->head);
    list->head = new_node;
    return (list->head != NULL) ? 0 : -1;
}

// returns 0 on success, -1 on failure
int remove_item(LinkedList *list, const int item_to_remove) {
    Node *marker = list->head;
    Node *temp = NULL;

    while (marker != NULL) {
        if (marker->value == item_to_remove) {
            if (temp == NULL) {
                if (marker->next == NULL) {
                    list->head = NULL;
                    destroy_node(marker);
                    marker = NULL;
                } else {
                    list->head = create_node(marker->value, marker->next);
                    destroy_node(marker);
                    marker = NULL;
                }
                return 0;
            } else {
                temp->next = marker->next;
                destroy_node(marker);
                temp = NULL;
                return 0;
            }
        }
        temp = marker;
        marker = marker->next;
    }
    return -1; // failure
}

void print_list(const LinkedList *list) {
    Node *marker = list->head;
    while (marker != NULL) {
        printf("%d\n", marker->value);
        marker = marker->next;
    }
}

int main(int argc, char **argv) {
    LinkedList *list = create_linkedlist();

    insert(list, 1);
    insert(list, 2);
    insert(list, 3);
    insert(list, 4);

    printf("The fully created list is:\n");
    print_list(list);
    printf("\nNow removing elements:\n");
    remove_item(list, 4);
    print_list(list);
    printf("\n");

    remove_item(list, 1);
    print_list(list);
    printf("\n");

    remove_item(list, 2);
    print_list(list);
    printf("\n");

    remove_item(list, 3);
    print_list(list);

    destroy_linkedlist(list);

    return 0;
}
```