CPSC 3125, Operating Systems Lab Assignment

Lab 07

1 Instructions

Using the starter code below, implement a linked list. You are given these four functions:
→ int menu();
→ void dowhat(int);
→ Node *create_ll(Node *);
→ Node *display(Node *);
You implemented these four functions last week:
→ Node *insert_beg(Node *);
→ Node *insert_end(Node *);
→ Node *insert_before(Node *);
→ Node *insert_after(Node *);
You will implement these five functions this week:
→ Node *delete_beg(Node *);
→ Node *delete_end(Node *);
→ Node *delete_node(Node *);
→ Node *delete_after(Node *);
→ Node *delete_list(Node *);

I have given you an executable you can use as a guide. This is a strenuous exercise, but you will learn a lot about pointer manipulatio.

2 Starter Code

```c
/**********************************************
 * Name: lab07.c
 * Author: C data structures
 * October 23, 2021
 * Purpose: Write a program to create a linked list and perform insertions and deletions of
    all cases.
 * Write functions to sort and finally delete the entire list at once.
 * Compile with: lab07.c -o lab07.exe -Wall
 * ******************************************/

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

struct node
{
    int data;
    struct node *next;
};

typedef struct node Node;

//variable declarations
int option;
Node *start = NULL;

```

```c
//function declarations
int menu();
void dowhat(int);

Node *create_ll(Node *);
Node *display(Node *);
Node *insert_beg(Node *);
Node *insert_end(Node *);
Node *insert_before(Node *);
Node *insert_after(Node *);

int main()
{
    system("clear");
    do
    {
        option = menu();
        printf("start = %p\n", start);
        dowhat(option);
    }
    while(option != 13);

    return 0;
}

int menu()
{
    printf("\n\n *****MAIN MENU *****");
    printf("\n 1: Create a list");
    printf("\n 2: Display the list");
    printf("\n 3: Add a node at the beginning");
    printf("\n 4: Add a node at the end");
    printf("\n 5: Add a node before a given node");
    printf("\n 6: Add a node after a given node");
    printf("\n 13: EXIT");
    int option;
    printf("\n\n Enter your option : ");
    scanf("%d", &option);

    return option;
}

void dowhat(int option)
{
    switch(option)
    {
        case 1: start = create_ll(start);
            printf("\n LINKED LIST CREATED");
            break;
        case 2: start = display(start);
            break;
        case 3: start = insert_beg(start);
            break;
        case 4: start = insert_end(start);
            break;
        case 5: start = insert_before(start);
            break;
        case 6: start = insert_after(start);
            break;
    }
}

Node *create_ll(Node *start)
{
    Node *new_node, *ptr;
    int num;
    printf(" Enter the data  (-1 to end): ");
    scanf("%d", &num);
    while(num != -1)
    {
        new_node = (Node*)malloc(sizeof(Node));
        new_node->data = num;
        if(start==NULL)
        {
            //printf("if branch, start == NULL\n");
            new_node->next = NULL;
            start = new_node;
        }
        else
        {
            //printf("else branch, start == %p\n", start);
            ptr = start;
            while(ptr->next!=NULL)
                ptr = ptr->next;
            ptr->next = new_node;
            new_node->next = NULL;
        }
        printf(" Enter the data : ");
        scanf("%d", &num);
    }
    return start;
}

Node *display(Node *start)
{
    Node *ptr;
    ptr = start;
    printf("\nLinked List ------------------------------\n");
    printf(" - start->%p\n", start);
    while(ptr != NULL)
    {
        printf(" - %p<-%d->%p\n", ptr, ptr->data, ptr->next);
        ptr = ptr->next;
    }
    printf("\n------------------------------------------\n");
    return start;
}

Node *insert_beg(Node *start)
{
    //declare a pointer to a ne Node
    //declare a new integer as a data value
    printf("\n Enter the data : ");
    //get user input for the data
    //malloc memory for a new Node
    //initialize the data member to the new integer
    //initialize the next member to the start of the list
    //set start to the address of the new Node
    printf("in insert_beg() --- new_node address: %p, new_node->data: %d, new_node->next: %p
        \n", new_node, new_node->data, new_node->next);
    return start;
}

Node *insert_end(Node *start)
{
    //declare a pointer to a new Node, and a pointer to a iteration node
    //declare a new integer as a data value
    printf(" Enter the data : ");
    //get user input for the data
    //malloc memory for a new Node
    //initialize the data member to the new integer
    //initialize the next member to NULL (the end of the list)
    //set the iteration pointer to start
    while(ptr->next != NULL)
        ptr = ptr->next;
    //set the iteration pointer next member to the new Node
```

```c
    printf("in insert_end() --- new_node_address: %p, new_node->data: %d, new_node->next: %p
        \n", new_node, new_node->data, new_node->next);
    return start;
}

Node *insert_before(Node *start)
{
    //declare a pointer to a new Node, a pointer to a iteration node, and a pointer to the
        ``pre'' insertion Node
    //declare a new integer as a data value, and a new integer to hold the value before
        which the new Node is to be inserted
    printf("\n Enter the data : ");
    //get the data value from the user
    printf("\n Enter the value before which the data has to be inserted : ");
    //get the ``before'' value from the user
    //malloc memory for a new Node
    //initialize the data member to the new integer
    //set the iteration pointer to start
    while(ptr->data != val)
    {
        preptr = ptr;
        ptr = ptr->next;
    }
    //set the next member of the pre-pointer to the new Node
    printf("in insert_before() --- new_node_address: %p, new_node->data: %d, new_node->next:
        %p\n", new_node, new_node->data, new_node->next);
    return start;
}

Node *insert_after(Node *start)
{
    //declare a pointer to a new Node, a pointer to a iteration node, and a pointer to the
        ``pre'' insertion Node
    //declare a new integer as a data value, and a new integer to hold the value after which
         the new Node is to be inserted
    printf("\n Enter the data : ");
    //get the data value from the user
    printf("\n Enter the value after which the data has to be inserted : ");
    //get the ``after'' value from the user
    //malloc memory for a new Node
    //initialize the data member to the new integer
    //set the iteration pointer to start
    //set the preptr to ptr
    while(preptr->data != val)
    {
        preptr = ptr;
        ptr = ptr->next;
    }
    //set the next member of the pre-pointer to the new Node
    //set the next member of the new Node to the pointer
    printf("in insert_after() --- new_node_address: %p, new_node->data: %d, new_node->next:
        %p\n", new_node, new_node->data, new_node->next);
    return start;
}

Node *delete_beg(Node *start)
{
    if(start == NULL)
    {
        printf("no list to delete\n");
        return NULL;
    }
    //declare a pointer to a Node object
    //set pointer to start
    //set start variable to the next member of the start Node
    //free the pointer
    return start;
}
```

```

Node *delete_end(Node *start)
{
    //declare to Node pointers, one to ptr and one to preptr
    //set ptr to start
    //iterate through list to end
    while(ptr->next != NULL)
    {
        preptr = ptr;
        ptr = ptr->next;
    }
    //set the next member of preptr to NULL
    //free the pointer
    return start;
}

Node *delete_node(Node *start)
{
    //declare variables to ptr and preptr
    //declare an integer variable
    printf("\n Enter the value of the node which has to be deleted : ");
    //read the input from the user
    //set ptr to start
    if(ptr->data == val) //the value to delete is the first item
    {
        start = delete_beg(start);
        return start;
    }
    else
    {
        while(ptr->data != val)
        {
            preptr = ptr;
            ptr = ptr->next;
        }
        //set the next member of preptr to tje next member of ptr
        //free the ptr
        return start;
    }
}

Node *delete_after(Node *start)
{
    //declare pointers to ptr and preptr
    //declare an integer variable
    printf("\n Enter the value after which the node has to deleted : ");
    //get input from the user
    //set ptr to start
    //set preptr to pstr
    while(preptr->data != val)
    {
        preptr = ptr;
        ptr = ptr->next;
    }
    //seet the next member of preptr to the next member of ptr
    //free the ptr
    return start;
}

Node *delete_list(Node *start)
{
    if(start == NULL)
    {
        printf("no list to delete\n");
        return NULL;
    }
    //declare a pointer to Node
    //set the ptr to start
    while(1)
    {
        //set start to the return value to the delete_bet function
        if(start == NULL)
        {
            printf("no list to delete\n");
            return NULL;
        }
        else
            //set ptr to start
    }
    return start;
}
```

3 Output

See the demonstration in class and the accompanying executable.

4 Lab deliverable

Your deliverable consists of (1) the C source code, and (2) a text document showing the output of the program.