#include <stdio.h>

#include <malloc.h>

typedef struct node Node;

struct node

{

        int data;

    Node \*next;

};

Node \*head = NULL, \*last;

int size(){

        Node \*i = head;

    int len = 0;

    while (i != NULL)

    {

            len++;

        i = i->next;

    }

    return len;

}

void addNodeAtStart(int data) {

        Node \*newNode = (Node\*)malloc(sizeof(Node));

    newNode->next = head;

    head = newNode;

    newNode->data = data;

}

void addNodeAtEnd(int data)

{

        Node \*newNode = (Node \*)malloc(sizeof(Node));

    newNode->data = data;

    if (head == NULL)

    {

            head = newNode;

        head->next = NULL;

        last = head;

    }

    else

    {

            last->next = newNode;

        last = newNode;

        newNode->next = NULL;

    }

}

void addNodeAt(int index, int data){

        if (index < 0){

            return;

    }

    if (index > size() - 1){

            return;

    }

    Node \*newNode = (Node \*)malloc(sizeof(Node));

    newNode->data = data;

    Node \*currNode = head, \*prev;

    int i;

    for (i = 0; i < index; i++)

    {

            prev = currNode;

        currNode = currNode->next;

    }

    newNode->next = currNode;

    prev->next = newNode;

}

void deleteAt(int index)

{

        if (index < 0){

            return;

    }

    if (index > size() - 1){

            return;

    }

    Node \*currNode = head, \*prev;

    int i;

    for (i = 0; i < index; i++) {

            prev = currNode;

        currNode = currNode->next;

    }

    prev->next = currNode->next;

    free(currNode);

}

void deleteFirst(){

        Node \*temp = head;

    head = head->next;

    free(temp);

}

void deleteLast(){

        Node \*temp = head, \*prev;

    while(temp->next != NULL){

            prev = temp;

        temp = temp->next;

    }

    last = prev;

    prev->next = NULL;

    free(temp);

}

void traverse(char message[])

{

        Node \*i = head;

    printf("LL (%s) : ", message);

    while (i != NULL)

    {

            printf("%d ", i->data);

        i = i->next;

    }

    printf("\n");

}

int main()

{

        addNodeAtEnd(1);

    addNodeAtEnd(3);

    addNodeAtEnd(4);

    addNodeAtEnd(12);

    addNodeAtEnd(123);

    traverse("node added at last");

    addNodeAt(3, 8);

    traverse("node added at index 3");

    addNodeAtStart(10);

    traverse("node added at head");

    deleteAt(2);

    traverse("node at index 2 deleted");

    deleteFirst();

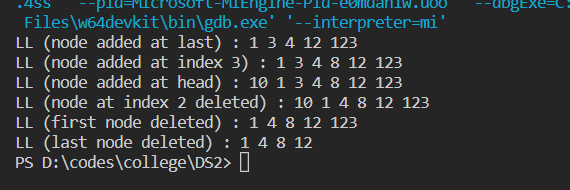
    traverse("first node deleted");

    deleteLast();

    traverse("last node deleted");

    return 0;

}



#include <stdio.h>

#include <malloc.h>

typedef struct node Node;

struct node {

    int data;

    Node \*next;

    Node \*prev;

};

Node \*head = NULL, \*last = NULL;

int size(){

    Node \*i = head;

    int len = 0;

    while (i != NULL)

    {

        len++;

        i = i->next;

    }

    return len;

}

void addNodeAtStart(int data) {

    Node \*newNode = (Node\*)malloc(sizeof(Node));

    newNode->data = data;

    head->prev = newNode;

    newNode->next = head;

    newNode->prev = NULL;

    head = newNode;

}

void addNodeAtEnd(int data){

    Node \*newNode = (Node\*)malloc(sizeof(Node));

    newNode->data = data;

    if(head == NULL){

        head = newNode;

        newNode->prev = NULL;

        newNode->next = NULL;

        last = newNode;

    } else {

        last->next = newNode;

        newNode->prev = last;

        newNode->next = NULL;

        last = newNode;

    }

}

void addNodeAt(int index, int data){

    if (index < 0){

        return;

    }

    if (index > size() - 1){

        return;

    }

    Node \*newNode = (Node \*)malloc(sizeof(Node));

    newNode->data = data;

    Node \*currNode = head, \*prev;

    for(int i = 0; i < index; i++){

        currNode = currNode->next;

    }

    newNode->prev = currNode->prev;

    newNode->next = currNode;

    currNode->prev->next = newNode;

    currNode->prev = newNode;

}

void deleteAt(int index){

    if (index < 0){

        return;

    }

    if (index > size() - 1){

        return;

    }

    Node \*currNode = head;

    for(int i = 0; i < index; i++){

        currNode = currNode->next;

    }

    currNode->prev->next = currNode->next;

    free(currNode);

}

void deleteFirst(){

    Node \*temp = head;

    head = head->next;

    head->prev = NULL;

    free(temp);

}

void deleteLast(){

    Node \*temp = last;

    last = last->prev;

    last->next = NULL;

    free(temp);

}

void traverse(char message[])

{

    Node \*i = head;

    printf("LL (%s) : [ ", message);

    while (i != NULL)

    {

        printf("%d ", i->data);

        i = i->next;

    }

    printf("]\n");

}

int search(int data){

    Node \*i = head;

    int index = 0;

    for(i = head; i != NULL; i=i->next){

        if(i->data == data){

            return index;

            break;

        }

        index++;

    }

    return -1;

}

int main(){

    addNodeAtEnd(10);

    addNodeAtEnd(20);

    addNodeAtEnd(30);

    addNodeAtEnd(40);

    addNodeAtEnd(50);

    addNodeAtEnd(60);

    traverse("node added at last");

    addNodeAt(3, 45);

    traverse("node added at index 3");

    addNodeAtStart(5);

    traverse("node added at head");

    deleteAt(2);

    traverse("node at index 2 deleted");

    deleteFirst();

    traverse("first node deleted");

    deleteLast();

    traverse("last node deleted");

    int i = search(45);

    printf("data = 45 at index : %d", i);

    return 0;

}

