AIM:DOUBLY LINKED LIST

ALGORTHIM

Step 1:if ptr=NULL

Step 2:set new node =ptr

Step 3:set ptr=ptr->NEXT

Step 4:set new node->data=val

Step 5:set new node->PREV=NULL

Step 6:set new node->next=start

Step 7:set head->prev=new node

Step 8:set head=new node

PROGRAM

#include <stdio.h>

#include <stdlib.h>

// Define the structure of the doubly linked list node

struct Node {

int data;

struct Node\* prev;

struct Node\* next;

};

// Function to create a new node and return its pointer

struct Node\* createNode(int data) {

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

newNode->data = data;

newNode->prev = NULL;

newNode->next = NULL;

return newNode;

}

// Function to insert a new node at the beginning of the linked list

void insertAtBeginning(struct Node\*\* head, int data) {

struct Node\* newNode = createNode(data);

if (\*head == NULL) {

\*head = newNode;

} else {

(\*head)->prev = newNode;

newNode->next = \*head;

\*head = newNode;

}

}

// Function to insert a new node at the end of the linked list

void insertAtEnd(struct Node\*\* head, int data) {

struct Node\* newNode = createNode(data);

if (\*head == NULL) {

\*head = newNode;

} else {

struct Node\* temp = \*head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = newNode;

newNode->prev = temp;

}

}

// Function to delete a node from the linked list

void deleteNode(struct Node\*\* head, struct Node\* node) {

if (\*head == NULL || node == NULL) {

return;

}

if (\*head == node) {

\*head = node->next;

}

if (node->next != NULL) {

node->next->prev = node->prev;

}

if (node->prev != NULL) {

node->prev->next = node->next;

}

free(node);

}

// Function to print the linked list

void printList(struct Node\* head) {

while (head != NULL) {

printf("%d -> ", head->data);

head = head->next;

}

printf("NULL\n");

}

int main() {

struct Node\* head = NULL;

insertAtBeginning(&head, 5);

insertAtBeginning(&head, 10);

insertAtEnd(&head, 15);

printList(head);

deleteNode(&head, head);

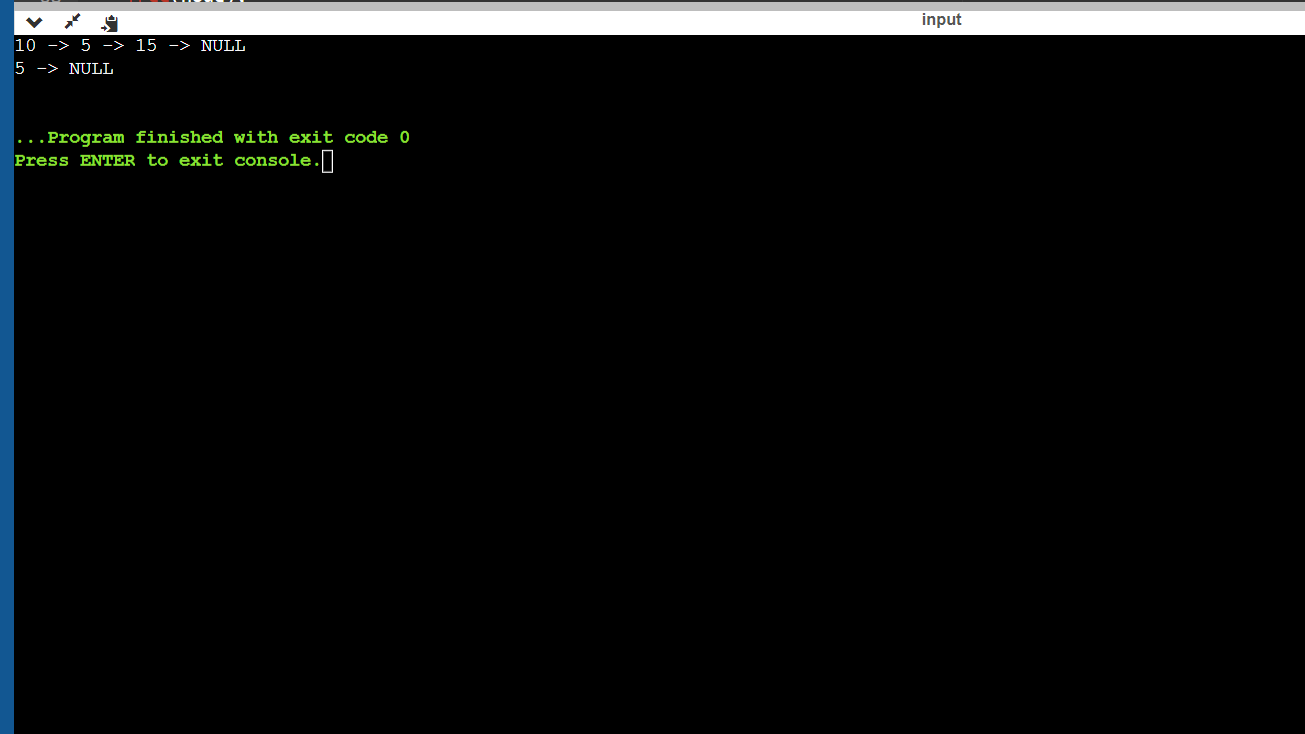
deleteNode(&head, head->next);

printList(head);

return 0;

}

OUTPUT:



GITHUB LINK