**ASSIGNMENT 4 – DUBLY LINKED LIST**

**ARGHA MALLICK – 11500122014**

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \*next;

struct Node \*prev;

};

struct Node \*createSLL(int value) {

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

newNode->data = value;

newNode->next = NULL;

newNode->prev = NULL;

return newNode;

}

void addAtBeginning(struct Node \*\*head, int value) {

struct Node \*newNode = createSLL(value);

newNode->next = \*head;

if (\*head != NULL) {

(\*head)->prev = newNode;

}

\*head = newNode;

}

void addAtEnd(struct Node \*\*head, int value) {

struct Node \*newNode = createSLL(value);

struct Node \*curr = \*head;

while (curr->next != NULL) {

curr = curr->next;

}

curr->next = newNode;

newNode->prev = curr;

}

void addAtIntermediate(struct Node \*prevNode, int value) {

if (prevNode == NULL) {

printf("Clouldn't add. The given node doesn't exist.\n");

return;

}

struct Node \*newNode = createSLL(value);

newNode->next = prevNode->next;

if (prevNode->next != NULL) {

prevNode->next->prev = newNode;

}

newNode->prev = prevNode;

prevNode->next = newNode;

}

void deleteFromBeginning(struct Node \*\*head) {

if (\*head == NULL) {

printf("List is empty!\n");

return;

}

struct Node \*temp = \*head;

\*head = (\*head)->next;

if (\*head != NULL) {

(\*head)->prev = NULL;

}

free(temp);

}

void deleteFromEnd(struct Node \*\*head) {

if (\*head == NULL) {

printf("List is Empty.\n");

return;

}

struct Node \*curr = \*head;

while (curr->next != NULL) {

curr = curr->next;

}

if (curr->prev != NULL) {

curr->prev->next = NULL;

} else {

\*head = NULL;

}

free(curr);

}

void deleteFromIntermediate(struct Node \*\*head, struct Node \*toDel) {

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

printf("Clouldn't delete. The given node doesn't exixt.\n");

return;

}

if (\*head == toDel) {

\*head = (\*head)->next;

}

if (toDel->next != NULL) {

toDel->next->prev = toDel->prev;

}

if (toDel->prev != NULL) {

toDel->prev->next = toDel->next;

}

free(toDel);

}

void displaySLL(struct Node \*head) {

struct Node \*curr = head;

printf("NULL <-> ");

while (curr != NULL) {

printf("%d <-> ", curr->data);

curr = curr->next;

}

printf("NULL\n");

}

int main() {

struct Node \*head = NULL;

int choise, value, prevValue;

do {

printf("<--:: MAIN MENU ::-->\n1. Add At Beginning\n2. Add At End\n3. "

"Display DLL\n4. Add At Intermediate\n5. Delete from Beginning\n6. "

"Delete From End\n7. Delete from Intermediate\n0. Exit\n\nENTER "

"YOUR CHOISE ::-->> ");

scanf("%d", &choise);

switch (choise) {

case 1:

printf("Enter the value: ");

scanf("%d", &value);

addAtBeginning(&head, value);

printf("CURRENT LINKED LIST\n");

displaySLL(head);

break;

case 2:

printf("Enter the value: ");

scanf("%d", &value);

addAtEnd(&head, value);

printf("CURRENT LINKED LIST\n");

displaySLL(head);

break;

case 3:

printf("CURRENT LINKED LIST\n");

displaySLL(head);

break;

case 4:

printf("Enter the value to be added: ");

scanf("%d", &value);

printf("Enter the value after which you want to add: ");

scanf("%d", &prevValue);

struct Node \*curr = head;

while (curr->data != prevValue) {

curr = curr->next;

}

struct Node \*prevNode = curr;

addAtIntermediate(prevNode, value);

printf("CURRENT LINKED LIST\n");

displaySLL(head);

break;

case 5:

deleteFromBeginning(&head);

printf("CURRENT LINKED LIST\n");

displaySLL(head);

break;

case 6:

deleteFromEnd(&head);

printf("CURRENT LINKED LIST\n");

displaySLL(head);

break;

case 7:

printf("Enter the value to be deleted: ");

scanf("%d", &value);

struct Node \*curr2 = head;

while (curr2->data != value) {

curr2 = curr2->next;

}

struct Node \*toDel = curr2;

deleteFromIntermediate(&head, toDel);

printf("CURRENT LINKED LIST\n");

displaySLL(head);

case 0:

break;

default:

printf("Invalid Choise! Try Again.\n");

break;

}

} while (choise != 0);

}

OUTPUT

