**CIRCULAR LINKED LIST**

#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

int data;

struct Node\* next;

} Node;

Node\* createNode(int data) {

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

newNode->data = data;

newNode->next = NULL;

return newNode;

}

Node\* createList(Node\* last, int data) {

Node\* newNode = createNode(data);

if (last == NULL) {

last = newNode;

last->next = last;

} else {

newNode->next = last->next;

last->next = newNode;

last = newNode;

}

return last;

}

Node\* insertAtBeginning(Node\* last, int data) {

if (last == NULL) return createList(last, data);

Node\* newNode = createNode(data);

newNode->next = last->next;

last->next = newNode;

return last;

}

Node\* insertAtEnd(Node\* last, int data) {

return createList(last, data);

}

Node\* insertAtPosition(Node\* last, int data, int position) {

if (last == NULL && position != 1){

return last;

}

if (position == 1){

return insertAtBeginning(last, data);

}

Node\* newNode = createNode(data);

Node\* temp = last->next;

for (int i = 1; i < position - 1; i++) {

if (temp->next == last->next) return last;

temp = temp->next;

}

newNode->next = temp->next;

temp->next = newNode;

if (temp == last) last = newNode;

return last;

}

Node\* deleteFirstNode(Node\* last) {

if (last == NULL) return NULL;

Node\* head = last->next;

if (last == head) {

free(head);

return NULL;

}

last->next = head->next;

free(head);

return last;

}

Node\* deleteLastNode(Node\* last) {

if (last == NULL) return NULL;

Node\* temp = last->next;

if (last == temp) {

free(last);

return NULL;

}

while (temp->next != last) temp = temp->next;

temp->next = last->next;

free(last);

return temp;

}

Node\* deleteSpecific(Node\* last, int value) {

if (last == NULL) return NULL;

Node\* temp = last->next, \*prev = last;

do {

if (temp->data == value) {

if (temp == last) last = deleteLastNode(last);

else {

prev->next = temp->next;

free(temp);

}

return last;

}

prev = temp;

temp = temp->next;

} while (temp != last->next);

return last;

}

void displayList(Node\* last) {

if (last == NULL) {

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

return;

}

Node\* temp = last->next;

do {

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

temp = temp->next;

} while (temp != last->next);

printf("\n");

}

int main() {

Node\* last = NULL;

int choice, data, position;

printf("\nShreya J G 1BM23IC061\n");

do {

printf("1.Create 2.Insert at Beginning 3.Insert at End 4.Insert at Position 5. Delete First Node 6.Delete Last Node 7.Delete Specific Element 8.Display List 9.Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter data: ");

scanf("%d", &data);

last = createList(last, data);

break;

case 2:

printf("Enter data: ");

scanf("%d", &data);

last = insertAtBeginning(last, data);

break;

case 3:

printf("Enter data: ");

scanf("%d", &data);

last = insertAtEnd(last, data);

break;

case 4:

printf("Enter data and position: ");

scanf("%d %d", &data, &position);

last = insertAtPosition(last, data, position);

break;

case 5:

last = deleteFirstNode(last);

break;

case 6:

last = deleteLastNode(last);

break;

case 7:

printf("Enter value to delete: ");

scanf("%d", &data);

last = deleteSpecific(last, data);

break;

case 8:

displayList(last);

break;

default:

printf("Invalid input\n");

}

} while (choice != 9);

return 0;

}

