***NAME : JANHAVI GATTANI***

***PRN : 12311291***

***ROLL NO:37***

***BATCH :2***

***LAB ASSIGNMENT 6***

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

struct Node\* insertAtEnd(struct Node\* head, int value);

struct Node\* insertAtBeginning(struct Node\* head, int value);

struct Node\* insertAtPosition(struct Node\* head, int value, int position);

struct Node\* deleteAtBeginning(struct Node\* head);

struct Node\* deleteAtEnd(struct Node\* head);

struct Node\* deleteAtPosition(struct Node\* head, int position);

void displayList(struct Node\* head);

int main() {

struct Node\* head = NULL;

int n, value, choice, position;

printf("Enter the number of elements in the linked list: ");

scanf("%d", &n);

printf("Enter the elements: ");

for (int i = 0; i < n; i++) {

scanf("%d", &value);

head = insertAtEnd(head, value);

}

while (1) {

printf("\nMenu:\n");

printf("1. Insert at Beginning\n");

printf("2. Insert at End\n");

printf("3. Insert at Position\n");

printf("4. Delete from Beginning\n");

printf("5. Delete from End\n");

printf("6. Delete from Position\n");

printf("7. Display List\n");

printf("8. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter value to insert: ");

scanf("%d", &value);

head = insertAtBeginning(head, value);

break;

case 2:

printf("Enter value to insert: ");

scanf("%d", &value);

head = insertAtEnd(head, value);

break;

case 3:

printf("Enter value to insert: ");

scanf("%d", &value);

printf("Enter position: ");

scanf("%d", &position);

head = insertAtPosition(head, value, position);

break;

case 4:

head = deleteAtBeginning(head);

break;

case 5:

head = deleteAtEnd(head);

break;

case 6:

printf("Enter position to delete: ");

scanf("%d", &position);

head = deleteAtPosition(head, position);

break;

case 7:

displayList(head);

break;

case 8:

exit(0);

default:

printf("Invalid choice!\n");

}

}

return 0;

}

struct Node\* insertAtBeginning(struct Node\* head, int value) {

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

newNode->data = value;

newNode->next = head;

return newNode;

}

struct Node\* insertAtEnd(struct Node\* head, int value) {

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

newNode->data = value;

newNode->next = NULL;

if (head == NULL) return newNode;

struct Node\* temp = head;

while (temp->next != NULL)

temp = temp->next;

temp->next = newNode;

return head;

}

struct Node\* insertAtPosition(struct Node\* head, int value, int position) {

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

newNode->data = value;

if (position == 1) {

newNode->next = head;

return newNode;

}

struct Node\* temp = head;

for (int i = 1; temp != NULL && i < position - 1; i++)

temp = temp->next;

if (temp == NULL) {

printf("Position out of range!\n");

return head;

}

newNode->next = temp->next;

temp->next = newNode;

return head;

}

struct Node\* deleteAtBeginning(struct Node\* head) {

if (head == NULL) {

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

return NULL;

}

struct Node\* temp = head;

head = head->next;

free(temp);

return head;

}

struct Node\* deleteAtEnd(struct Node\* head) {

if (head == NULL) {

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

return NULL;

}

if (head->next == NULL) {

free(head);

return NULL;

}

struct Node\* temp = head;

while (temp->next->next != NULL)

temp = temp->next;

free(temp->next);

temp->next = NULL;

return head;

}

struct Node\* deleteAtPosition(struct Node\* head, int position) {

if (head == NULL) {

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

return NULL;

}

if (position == 1) {

struct Node\* temp = head;

head = head->next;

free(temp);

return head;

}

struct Node\* temp = head;

struct Node\* prev = NULL;

for (int i = 1; temp != NULL && i < position; i++) {

prev = temp;

temp = temp->next;

}

if (temp == NULL) {

printf("Position out of range!\n");

return head;

}

prev->next = temp->next;

free(temp);

return head;

}

void displayList(struct Node\* head) {

if (head == NULL) {

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

return;

}

struct Node\* temp = head;

printf("Linked List: ");

while (temp != NULL) {

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

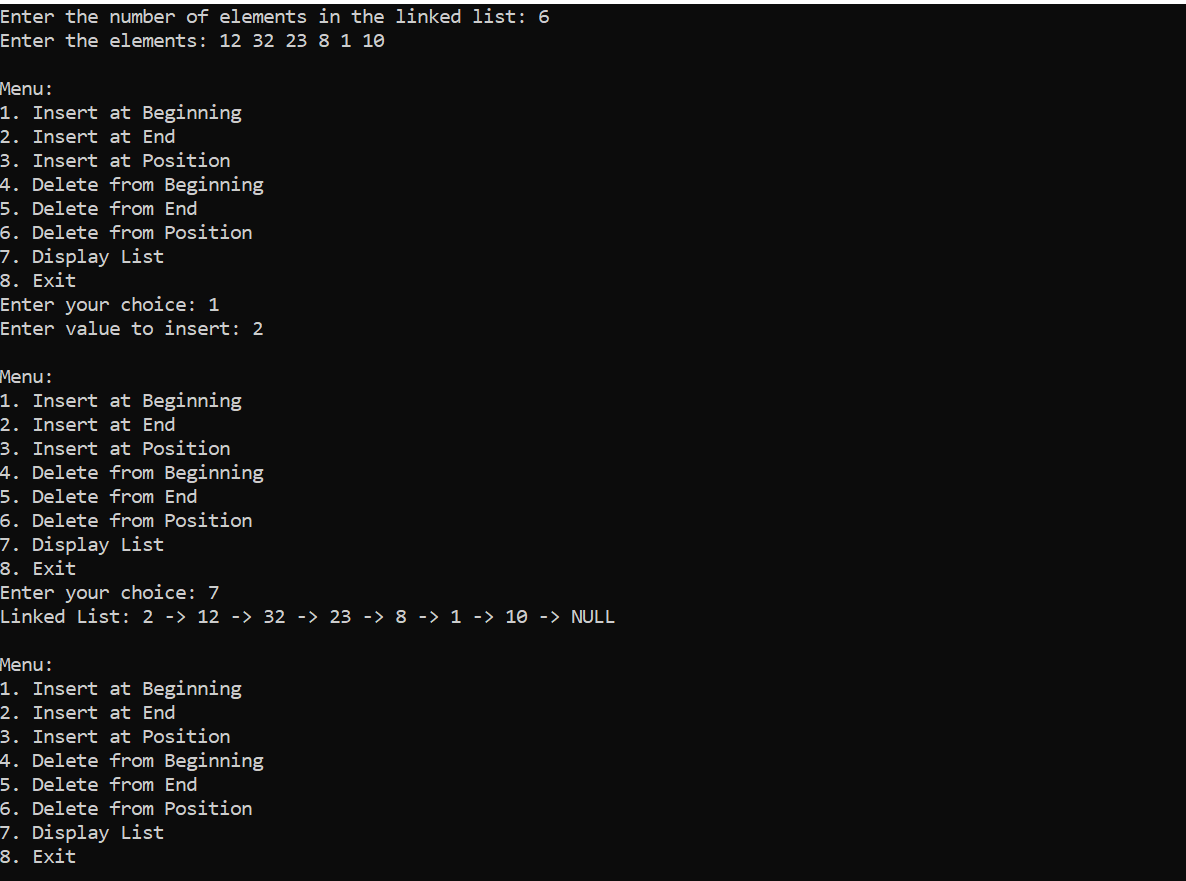
temp = temp->next;

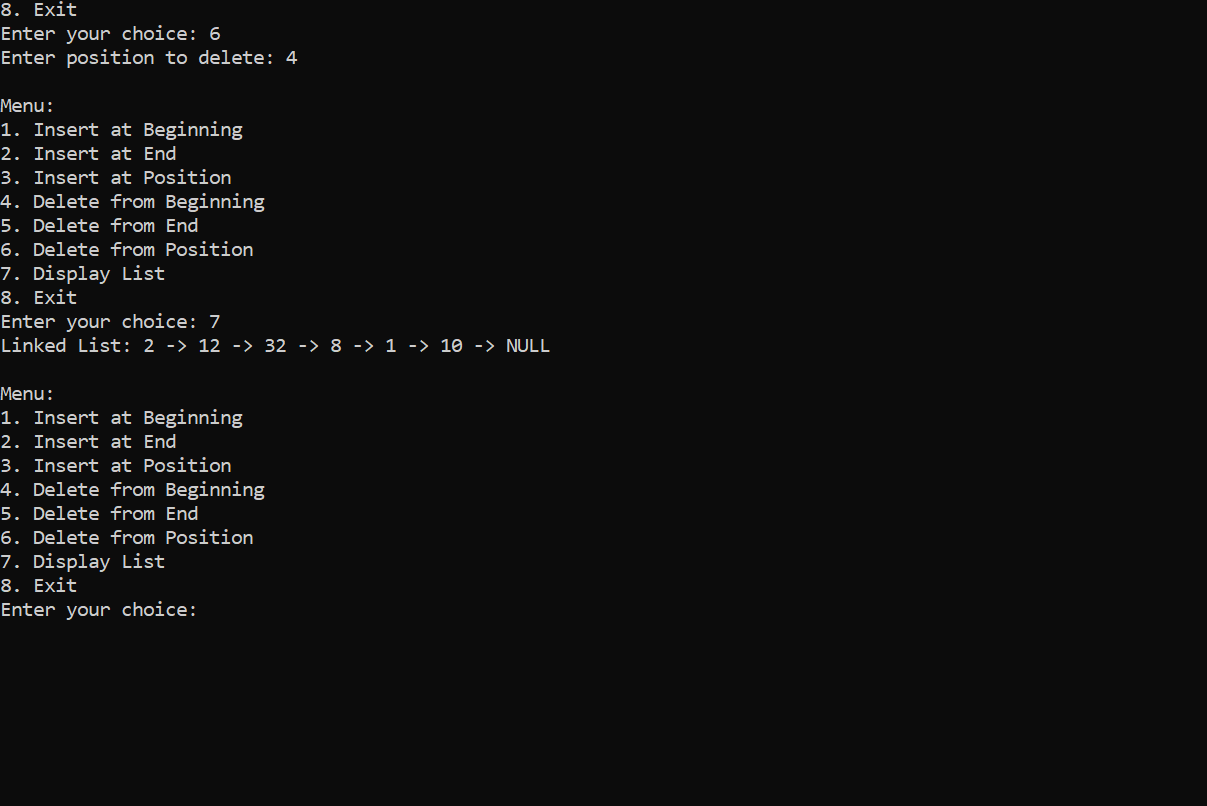
}

printf("NULL\n");

}

OUTPUT:





***DOUBLE LINKED LIST***

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* prev;

struct Node\* next;

};

struct Node\* insertAtEnd(struct Node\* head, int value);

struct Node\* insertAtBeginning(struct Node\* head, int value);

struct Node\* insertAtPosition(struct Node\* head, int value, int position);

struct Node\* deleteAtBeginning(struct Node\* head);

struct Node\* deleteAtEnd(struct Node\* head);

struct Node\* deleteAtPosition(struct Node\* head, int position);

void displayList(struct Node\* head);

int main() {

struct Node\* head = NULL;

int n, value, choice, position;

printf("Enter the number of elements in the doubly linked list: ");

scanf("%d", &n);

printf("Enter the elements: ");

for (int i = 0; i < n; i++) {

scanf("%d", &value);

head = insertAtEnd(head, value);

}

while (1) {

printf("\nMenu:\n");

printf("1. Insert at Beginning\n");

printf("2. Insert at End\n");

printf("3. Insert at Position\n");

printf("4. Delete from Beginning\n");

printf("5. Delete from End\n");

printf("6. Delete from Position\n");

printf("7. Display List\n");

printf("8. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter value to insert: ");

scanf("%d", &value);

head = insertAtBeginning(head, value);

break;

case 2:

printf("Enter value to insert: ");

scanf("%d", &value);

head = insertAtEnd(head, value);

break;

case 3:

printf("Enter value to insert: ");

scanf("%d", &value);

printf("Enter position: ");

scanf("%d", &position);

head = insertAtPosition(head, value, position);

break;

case 4:

head = deleteAtBeginning(head);

break;

case 5:

head = deleteAtEnd(head);

break;

case 6:

printf("Enter position to delete: ");

scanf("%d", &position);

head = deleteAtPosition(head, position);

break;

case 7:

displayList(head);

break;

case 8:

exit(0);

default:

printf("Invalid choice!\n");

}

}

return 0;

}

struct Node\* insertAtBeginning(struct Node\* head, int value) {

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

newNode->data = value;

newNode->prev = NULL;

newNode->next = head;

if (head != NULL) head->prev = newNode;

return newNode;

}

struct Node\* insertAtEnd(struct Node\* head, int value) {

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

newNode->data = value;

newNode->next = NULL;

if (head == NULL) {

newNode->prev = NULL;

return newNode;

}

struct Node\* temp = head;

while (temp->next != NULL)

temp = temp->next;

temp->next = newNode;

newNode->prev = temp;

return head;

}

struct Node\* insertAtPosition(struct Node\* head, int value, int position) {

if (position == 1)

return insertAtBeginning(head, value);

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

newNode->data = value;

struct Node\* temp = head;

for (int i = 1; temp != NULL && i < position - 1; i++)

temp = temp->next;

if (temp == NULL) {

printf("Position out of range!\n");

return head;

}

newNode->next = temp->next;

newNode->prev = temp;

if (temp->next != NULL)

temp->next->prev = newNode;

temp->next = newNode;

return head;

}

struct Node\* deleteAtBeginning(struct Node\* head) {

if (head == NULL) {

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

return NULL;

}

struct Node\* temp = head;

head = head->next;

if (head != NULL) head->prev = NULL;

free(temp);

return head;

}

struct Node\* deleteAtEnd(struct Node\* head) {

if (head == NULL) {

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

return NULL;

}

struct Node\* temp = head;

while (temp->next != NULL)

temp = temp->next;

if (temp->prev != NULL)

temp->prev->next = NULL;

else

head = NULL;

free(temp);

return head;

}

struct Node\* deleteAtPosition(struct Node\* head, int position) {

if (head == NULL) {

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

return NULL;

}

if (position == 1)

return deleteAtBeginning(head);

struct Node\* temp = head;

for (int i = 1; temp != NULL && i < position; i++)

temp = temp->next;

if (temp == NULL) {

printf("Position out of range!\n");

return head;

}

if (temp->next != NULL)

temp->next->prev = temp->prev;

if (temp->prev != NULL)

temp->prev->next = temp->next;

free(temp);

return head;

}

void displayList(struct Node\* head) {

if (head == NULL) {

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

return;

}

struct Node\* temp = head;

printf("Doubly Linked List: ");

while (temp != NULL) {

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

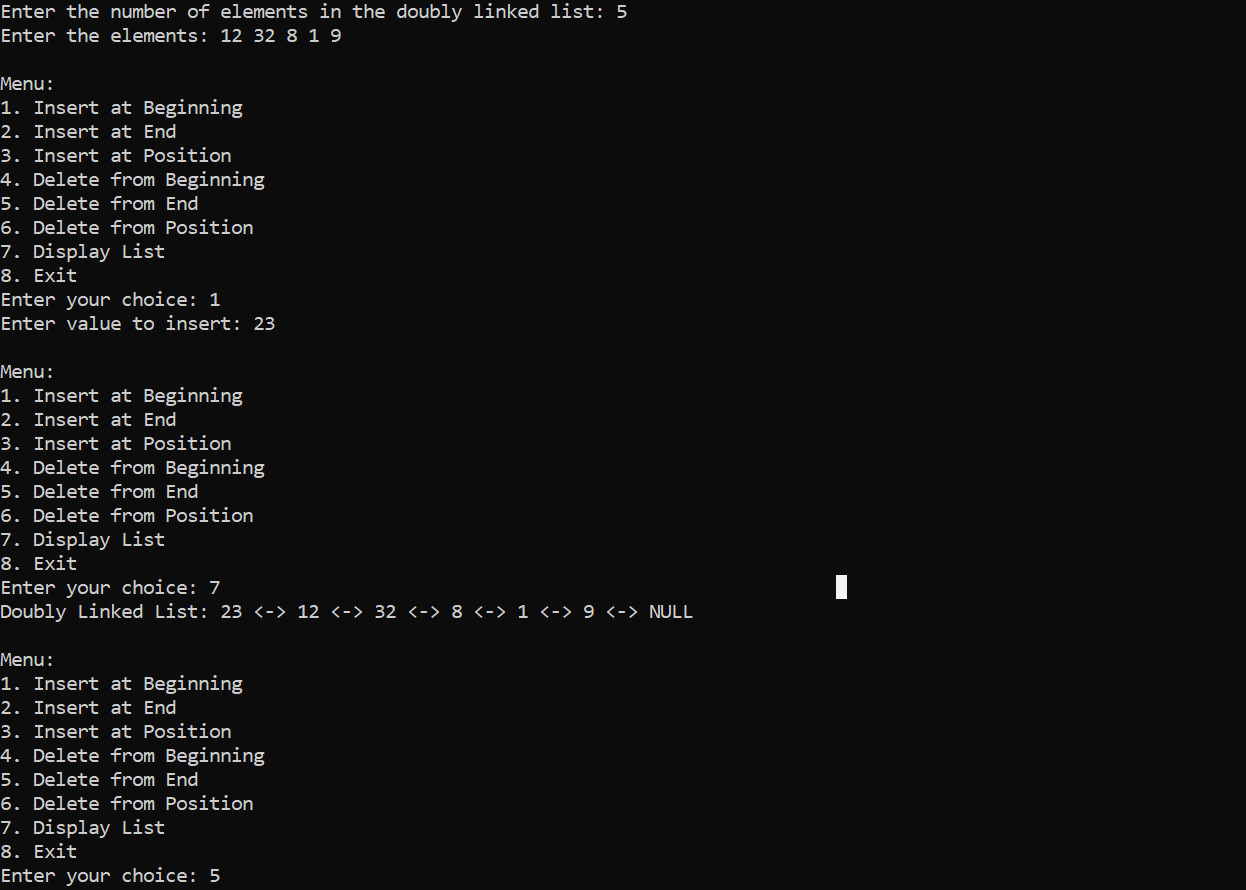
temp = temp->next;

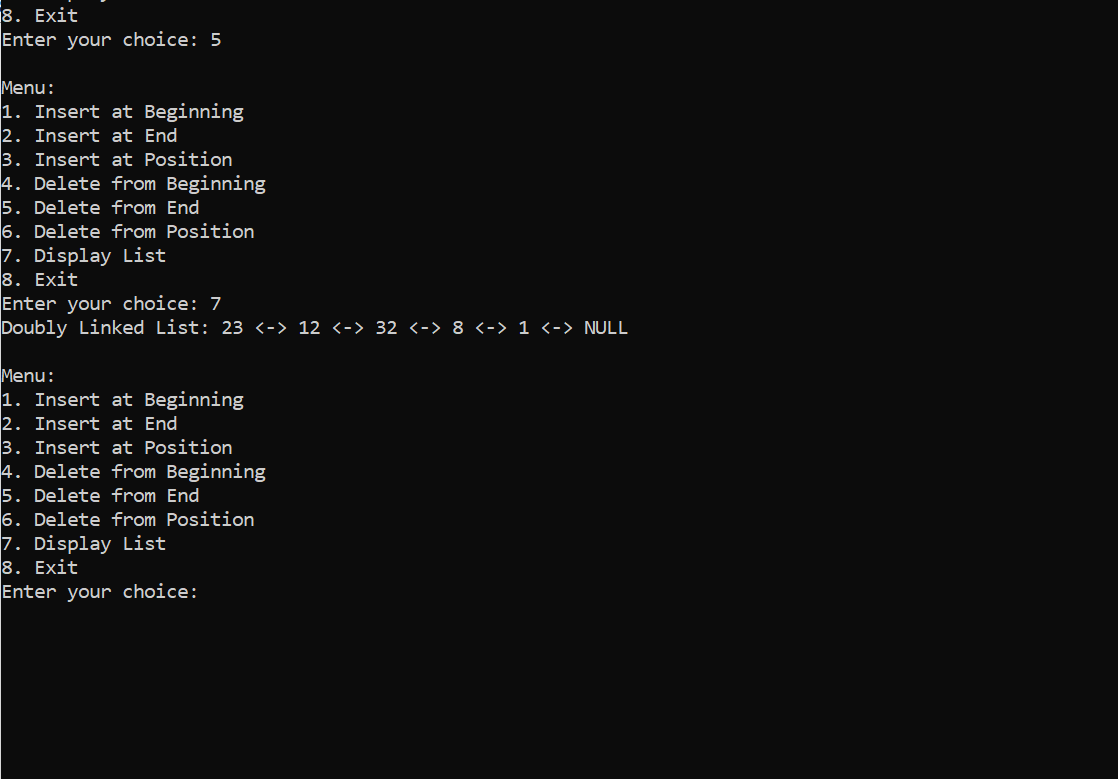
}

printf("NULL\n");

}

OUTPUT:





***CIRCULAR LINKED LIST***

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

struct Node\* insertAtBeginning(struct Node\* head, int value);

struct Node\* insertAtEnd(struct Node\* head, int value);

struct Node\* insertAtPosition(struct Node\* head, int value, int position);

struct Node\* deleteAtBeginning(struct Node\* head);

struct Node\* deleteAtEnd(struct Node\* head);

struct Node\* deleteAtPosition(struct Node\* head, int position);

void displayList(struct Node\* head);

int main() {

struct Node\* head = NULL;

int n, value, choice, position;

printf("Enter the number of elements in the circular linked list: ");

scanf("%d", &n);

printf("Enter the elements: ");

for (int i = 0; i < n; i++) {

scanf("%d", &value);

head = insertAtEnd(head, value);

}

while (1) {

printf("\nMenu:\n");

printf("1. Insert at Beginning\n");

printf("2. Insert at End\n");

printf("3. Insert at Position\n");

printf("4. Delete from Beginning\n");

printf("5. Delete from End\n");

printf("6. Delete from Position\n");

printf("7. Display List\n");

printf("8. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter value to insert: ");

scanf("%d", &value);

head = insertAtBeginning(head, value);

break;

case 2:

printf("Enter value to insert: ");

scanf("%d", &value);

head = insertAtEnd(head, value);

break;

case 3:

printf("Enter value to insert: ");

scanf("%d", &value);

printf("Enter position: ");

scanf("%d", &position);

head = insertAtPosition(head, value, position);

break;

case 4:

head = deleteAtBeginning(head);

break;

case 5:

head = deleteAtEnd(head);

break;

case 6:

printf("Enter position to delete: ");

scanf("%d", &position);

head = deleteAtPosition(head, position);

break;

case 7:

displayList(head);

break;

case 8:

exit(0);

default:

printf("Invalid choice!\n");

}

}

return 0;

}

struct Node\* insertAtBeginning(struct Node\* head, int value) {

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

newNode->data = value;

if (head == NULL) {

newNode->next = newNode;

return newNode;

}

struct Node\* temp = head;

while (temp->next != head)

temp = temp->next;

temp->next = newNode;

newNode->next = head;

return newNode;

}

struct Node\* insertAtEnd(struct Node\* head, int value) {

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

newNode->data = value;

if (head == NULL) {

newNode->next = newNode;

return newNode;

}

struct Node\* temp = head;

while (temp->next != head)

temp = temp->next;

temp->next = newNode;

newNode->next = head;

return head;

}

struct Node\* insertAtPosition(struct Node\* head, int value, int position) {

if (position == 1)

return insertAtBeginning(head, value);

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

newNode->data = value;

struct Node\* temp = head;

for (int i = 1; temp->next != head && i < position - 1; i++)

temp = temp->next;

newNode->next = temp->next;

temp->next = newNode;

return head;

}

struct Node\* deleteAtBeginning(struct Node\* head) {

if (head == NULL) {

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

return NULL;

}

struct Node\* temp = head, \*last = head;

while (last->next != head)

last = last->next;

if (head == last) {

free(head);

return NULL;

}

last->next = head->next;

head = head->next;

free(temp);

return head;

}

struct Node\* deleteAtEnd(struct Node\* head) {

if (head == NULL) {

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

return NULL;

}

struct Node\* temp = head, \*prev = NULL;

while (temp->next != head) {

prev = temp;

temp = temp->next;

}

if (head == temp) {

free(head);

return NULL;

}

prev->next = head;

free(temp);

return head;

}

struct Node\* deleteAtPosition(struct Node\* head, int position) {

if (head == NULL) {

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

return NULL;

}

if (position == 1)

return deleteAtBeginning(head);

struct Node\* temp = head, \*prev = NULL;

for (int i = 1; temp->next != head && i < position; i++) {

prev = temp;

temp = temp->next;

}

if (temp->next == head) {

printf("Position out of range!\n");

return head;

}

prev->next = temp->next;

free(temp);

return head;

}

void displayList(struct Node\* head) {

if (head == NULL) {

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

return;

}

struct Node\* temp = head;

printf("Circular Linked List: ");

do {

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

temp = temp->next;

} while (temp != head);

printf("(head)\n");

}

OUTPUT:

