**Practical-07**

Q. Implement a Circular Single Linked List (CSLL) and perform the operations: Create, Traverse, Insert\_Beg, Insert\_End, Delete\_beg, Delete\_end using Menu Driver Program.

Program:

//code for circular linked list

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \*next;

};

struct Node \*head = NULL;

void create();

void traverse();

void insert\_beg();

void insert\_end();

void delete\_beg();

void delete\_end();

int main() {

int choice;

do {

printf("\nMenu:\n");

printf("1. Create List\n");

printf("2. Traverse List\n");

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

printf("4. Insert at End\n");

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

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

printf("7. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

create();

printf("List created successfully.\n");

break;

case 2:

printf("Current list: ");

traverse();

break;

case 3:

insert\_beg();

printf("List after insertion at beginning: ");

traverse();

break;

case 4:

insert\_end();

printf("List after insertion at end: ");

traverse();

break;

case 5:

delete\_beg();

printf("List after deletion from beginning: ");

traverse();

break;

case 6:

delete\_end();

printf("List after deletion from end: ");

traverse();

break;

case 7:

printf("Exiting...\n");

break;

default:

printf("Invalid choice. Please try again.\n");

}

} while (choice != 7);

return 0;

}

void create() {

struct Node \*p;

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

printf("Enter the data of the first node: ");

scanf("%d", &p->data);

head = p;

p->next = head;

char ch;

do {

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

printf("Enter the data of next node: ");

scanf("%d", &q->data);

p->next = q;

q->next = head;

p = q;

printf("Press 'y' to continue or any other key to stop: ");

getchar();

ch = getchar();

} while (ch == 'y');

}

void traverse() {

if (head == NULL) {

printf("The list is empty.\n");

return;

}

struct Node \*p = head;

do {

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

p = p->next;

} while (p != head);

printf("(head)\n");

}

void insert\_beg() {

int value;

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

if (q == NULL) {

printf("Memory allocation failed!\n");

return;

}

printf("Enter the value for the new node: ");

scanf("%d", &value);

q->data = value;

if (head == NULL) {

head = q;

q->next = head;

} else {

struct Node \*p = head;

while (p->next != head) {

p = p->next;

}

p->next = q;

q->next = head;

head = q;

}

}

void insert\_end() {

int value;

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

if (q == NULL) {

printf("Memory allocation failed!\n");

return;

}

printf("Enter the value for the new node: ");

scanf("%d", &value);

q->data = value;

if (head == NULL) {

head = q;

q->next = head;

} else {

struct Node \*p = head;

while (p->next != head) {

p = p->next;

}

p->next = q;

q->next = head;

}

}

void delete\_beg() {

if (head == NULL) {

printf("The list is empty. Nothing to delete.\n");

return;

}

struct Node \*q = head;

if (head->next == head) {

free(head);

head = NULL;

} else {

struct Node \*p = head;

while (p->next != head) {

p = p->next;

}

p->next = head->next;

free(head);

head = p->next;

}

}

void delete\_end() {

if (head == NULL) {

printf("The list is empty. Nothing to delete.\n");

return;

}

if (head->next == head) {

free(head);

head = NULL;

return;

}

struct Node \*p = head;

struct Node \*q = NULL;

while (p->next != head) {

q = p;

p = p->next;

}

q->next = head;

free(p);

}

OUPUT:





