**Aim:** Implement Circular Linked List ADT.

#include <stdio.h> #include <stdlib.h>

struct Node {

int data;

struct Node \*next;

};

struct Node\* createNode(int value);

struct Node\* insertAtPosition(struct Node \*last, int data, int pos) {

if (last == NULL) {

if (pos != 1) {

printf("Invalid position!\n"); return last;

}

struct Node \*newNode = createNode(data); last = newNode;

last->next = last; return last;

}

struct Node \*newNode = createNode(data);

struct Node \*curr = last->next; if (pos == 1) {

newNode->next = curr; last->next = newNode; return last;

}

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

curr = curr->next;

if (curr == last->next) { printf("Invalid position!\n"); return last;

}

}

newNode->next = curr->next;

curr->next = newNode;

if (curr == last) last = newNode;

return last;

}

void printList(struct Node \*last) {

if (last == NULL) return;

struct Node \*head = last->next; while (1) {

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

if (head == last->next) break;

}

printf("\n");

}

struct Node\* createNode(int value) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node)); newNode->data = value;

newNode->next = NULL; return newNode;

}

int main() {

struct Node \*first = createNode(2); first->next = createNode(3);

first->next->next = createNode(4); struct Node \*last = first->next->next; last->next = first;

printf("Original list: "); printList(last);

int data = 5, pos = 2;

last = insertAtPosition(last, data, pos); printf("List after insertions: "); printList(last);

return 0;

}