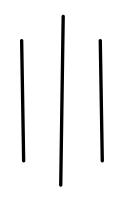
Tribhuvan University Institute Of Science and Technology

Prithvi Narayan Campus BSc.CSIT Program



PRACTICAL REPORT

(Data Structure and Algorithm)



Submitted To

Mr. Prithvi Raj Paneru

Department of Computer Science & Information Technology Prithvi Narayan Campus, Pokhara

Submitted By

Mahesh Kumar Udas Roll No. 21 2080 Batch

INDEX

Name: Mahesh Kumar Udas Roll No.: 21

Faculty: BSc.CSIT Semester: Third

Subject: Data Structure and Algorithm Year : Second

S.N.	Title	Date of Submission	Sign

OBJECTIVE: To implement Singly Linked List and perform various operations in C programming.

THEORY:

C++ A singly linked list is a linear data structure that stores data in nodes that are linked together in a chain. Each node has a value and a pointer to the next node in the list

Singly Linked List



Operations to perform in single linked list are mention bellow:

- 1. Insert Operation
 - a) Insert at beginning
 - b) Insert at Position
 - c) Insert at End

- 2. Delete Operation
 - a) Delete at beginning
 - b) Delete at position
 - c) Delere at End
- 3. Traverse Operation
 - a) Display items

PROGRAMS

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
// Single Linked List structure...
struct Node {
    int data:
    struct Node *next;
}; typedef struct Node node;
node *head = NULL:
// Function to create new node...
node* createNewNode(int item) {
    node *newnode =
(node*)malloc(sizeof(node));
    newnode -> data = item;
    newnode -> next = NULL;
    return newnode:
// Function to insert at beginning
void insertAtBeg(int item) {
    node *newnode = createNewNode(item);
    newnode -> next = head;
    head = newnode:
```

```
// Function to insert at a position
void insertAtPos(int pos, int item) {
     if (pos < 1) {
         printf("\n\t\!!! Invalid Position...\n");
         return;
     if (pos == 1) {
          insertAtBeg(item);
          return;
     node *temp = head;
     node *newnode = createNewNode(item);
    for (int i = 1; i < pos - 1 && temp! =
NULL; i++)
          temp = temp -> next;
     if (temp != NULL) {
          newnode -> next = temp -> next;
          temp->next = newnode;
     } else {
         printf("\n\t\t!!! Invalid Position...\n");
// Function to insert at the end
```

void insertAtEnd(int item) {

```
node *newnode = createNewNode(item);
                                                           if (head == NULL)  {
     if (head == NULL) head = newnode;
                                                                printf("\t!!! Empty Node list...\n");
     node *temp = head;
     while (temp->next != NULL)
                                                           while (temp != NULL)  {
                                                                printf("\%d \rightarrow ", temp->data);
          temp = temp -> next;
     temp->next = newnode;
                                                                temp = temp -> next;
}
                                                          printf("NULL\n");
// Function to delete at the beginning
void deleteAtBeg(){
     if(head==NULL) printf("\n\t!!!Empty
                                                     // Menu-driven Dashboard
Node list...\n'');
                                                      void Dashboard() {
     else\ head = head -> next;
                                                           int choice, value, pos;
                                                           do {
                                                                system("cls");
// Function to delete at position
void deleteAtPos(int pos){
                                                                display();
                                                                printf("\n\tenter operation: \n");
     if (pos < 1) {
                                                                printf("\t\t1. Insert at Beginning\n");
          printf("\n\t\t!!! Invalid Position...\n");
                                                                printf("\t\t2. Insert at Position\n");
                                                                printf("\t 3. Insert at End\n");
                                                                printf("\t\t4. Delete at Beginning\n");
     if (pos == 1) {
                                                                printf("\t\t5. Delete at Position\n");
          deleteAtBeg();
                                                                printf("\t\t6. Delete at End\n");
          return;
                                                                printf("\t\t7. Exit\n\n");
                                                                printf("\tEnter your choice: ");
     node *temp = head;
                                                                scanf("%d", &choice);
    for (int i = 1; i < pos - 1 && temp !=
                                                                switch (choice) {
NULL; i++)
                                                                     case 1:
                                                                          printf("\n\tEnter value to
          temp = temp -> next;
                                                      insert: ");
     if (temp != NULL)
                                                                          scanf("%d", &value);
          temp->next = (temp->next)->next;
                                                                          insertAtBeg(value);
                                                                          break:
     else {
                                                                     case 2:
          printf("\n\t\t!!! Invalid Position...\n");
                                                                          printf("\n\tEnter position: ");
}
                                                                          scanf("%d", &pos);
                                                                          printf("\tEnter value to
// Function to delete at end
                                                      insert: ");
void deleteAtEnd(){
                                                                          scanf("%d", &value);
     if(head==NULL) printf("\n\t!!!Empty
                                                                          insertAtPos(pos, value);
Node list...\n'');
                                                                          break;
     else{
                                                                     case 3:
          node *temp = head;
                                                                          printf("\n\tEnter value to
          while((temp->next)->next!=NULL)
                                                      insert: ");
               temp = temp -> next;
                                                                          scanf("%d", &value);
          temp->next = NULL;
                                                                          insertAtEnd(value);
                                                                          break;
                                                                     case 4:
                                                                          deleteAtBeg();
// Function to display the linked list
                                                                          break;
void display() {
                                                                     case 5:
     node *temp = head;
                                                                          printf("\n\tEnter position: ");
     printf("\n\tList\ of\ data:\n\t");
                                                                          scanf("%d", &pos);
```

```
deleteAtPos(pos);
                     break:
                                                                 printf("\n\tPress Enter to continue...");
                                                                 getch();
               case 6:
                    deleteAtEnd();
                                                            } while (choice != 7);
                     break:
               case 7:
                    printf("\n\tExiting
                                                       int main() {
                                                              Dashboard();
program... \n'');
                                                            return 0;
                     break;
               default:
                     printf("\n\tInvalid choice!
Please try again. n'');
```

```
Select C:\Users\Mahesh\Desktop\Linklist in C\output\SingleLinkedList.exe

List of data:

15 -> 20 -> 5 -> 10 -> 100 -> 50 -> 60 -> NULL

Enter operation:

1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Delete at Beginning
5. Delete at Position
6. Delete at End
7. Exit

Enter your choice: _____
```

```
C:\Users\Mahesh\Desktop\Linklist in C\output\SingleLinkedList.exe

List of data:

15 -> 20 -> 5 -> 10 -> 100 -> 50 -> 60 -> NULL

Enter operation:

1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Delete at Beginning
5. Delete at Position
6. Delete at End
7. Exit

Enter your choice: 1

Enter value to insert: 73

Press Enter to continue...
```

```
C:\Users\Mahesh\Desktop\Linklist in C\output\SingleLinkedList.exe

List of data:
73 -> 15 -> 20 -> 5 -> 10 -> 100 -> 50 -> 60 -> NULL

Enter operation:

1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Delete at Beginning
5. Delete at Position
6. Delete at End
7. Exit

Enter your choice: 5

Enter position: 7

Press Enter to continue...
```

```
C:\Users\Mahesh\Desktop\Linklist in C\output\SingleLinkedList.exe

List of data:

73 -> 15 -> 20 -> 5 -> 10 -> 100 -> 60 -> NULL

Enter operation:

1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Delete at Beginning
5. Delete at Beginning
6. Delete at End
7. Exit

Enter your choice: 7

Exiting program...

Press Enter to continue...
```

RESULTS AND DISCUSSION:

The stuents are successful to write code for single linked list in C programming. This program helps to understand basics of Data structure. The program have menu driven functioality to perform operations in single linked list.

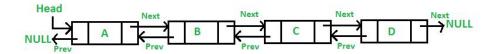
CONCLUSION:

This laboratory exercise provided a hands-on experience in DSA. Students gained practical knowledge of implementing basic in single linked list and now better equipped to undertake more complex programming tasks in the future.

OBJECTIVE: To implement Doubly Linked List and perform various operations in C programming.

THEORY:

A doubly linked list is a data structure that consists of nodes that are linked together in both directions. Each node has three parts: data, a pointer to the next node, and a pointer to the previous node.



Operations to perform in single linked list are mention bellow:

- 1. Insert Operation
 - a) Insert at beginning
 - b) Insert at Position
 - c) Insert at End

- 2. Delete Operation
 - a) Delete at beginning
 - b) Delete at position
 - c) Delere at End
- 3. Traverse Operation
 - a) Display items

PROGRAM:

#include<stdio.h>

```
#include<stdlib.h>
#include < conio. h >
// Doubly Linked List structure...
struct Node {
  int data:
  struct Node *prev:
  struct Node *next:
}; typedef struct Node node;
node *head = NULL;
// Function to create new node...
node* createNewNode(int item) {
  node *newnode =
(node*)malloc(sizeof(node));
  newnode -> data = item;
  newnode -> prev = NULL;
  newnode -> next = NULL;
  return newnode;
// Function to insert at beginning
void insertAtBeg(int item) {
  node *newnode = createNewNode(item);
  if (head != NULL) {
     newnode -> next = head;
```

```
head->prev = newnode;
  head = newnode:
// Function to insert at a specific position
void insertAtPos(int pos, int item) {
  if (pos < 1) {
    printf("\n\t\!!! Invalid Position...\n");
    return;
  if (pos == 1) {
    insertAtBeg(item);
    return;
  node *temp = head;
  node *newnode = createNewNode(item);
  for (int i = 1; i < pos - 1 && temp != NULL;
i++)
    temp = temp -> next;
  if (temp != NULL) {
    newnode->next = temp->next;
    if (temp->next != NULL)
       temp->next->prev=newnode;
    temp->next = newnode;
```

```
// Function to delete at end
     newnode -> prev = temp;
  } else {
                                                      void deleteAtEnd(){
    printf("\n\t\t!!! Invalid Position...\n");
                                                        if(head == NULL) printf("\n\t!!Empty Node")
                                                      list...\langle n''\rangle;
                                                        else {
                                                           node *temp = head;
// Function to insert at the end
                                                           while (temp->next != NULL)
void insertAtEnd(int item) {
                                                             temp = temp -> next;
  node *newnode = createNewNode(item);
                                                           if (temp->prev != NULL)
  if (head == NULL)  {
                                                             temp->prev->next=NULL;
     head = newnode;
                                                           else
                                                             head = NULL;
     return;
  node *temp = head;
  while (temp->next != NULL)
     temp = temp -> next;
                                                      // Function to display the linked list
  temp->next = newnode;
                                                      void display() {
  newnode -> prev = temp;
                                                        node *temp = head;
                                                        printf("\n\tList\ of\ data:\n\t");
                                                        if (head == NULL)  {
                                                           printf("\t!!! Empty Node list...\n");
// Function to delete at the beginning
void deleteAtBeg(){
                                                           return;
  if(head == NULL) printf("\n\t!!!Empty Node
list...\langle n''\rangle;
                                                        while (temp != NULL)  {
  else {
                                                           printf("\%d <->", temp->data);
     head = head -> next;
                                                           temp = temp -> next;
     if (head != NULL)
       head -> prev = NULL;
                                                        printf("NULL\n");
                                                      // Menu-driven Dashboard
// Function to delete at position
                                                      void Dashboard() {
                                                        int choice, value, pos;
void deleteAtPos(int pos){
  if (pos < 1) {
                                                        do {
     printf("\n\t\!!! Invalid Position...\n");
                                                           system("cls");
     return;
                                                           display();
  if (pos == 1) \{
                                                           printf("\n\tEnter operation: \n");
                                                           printf("\t\t1. Insert at Beginning\n");
     deleteAtBeg();
                                                           printf("\t\t2. Insert at Position\n");
     return;
                                                           printf("\t\t3. Insert at End\n");
                                                           printf("\t\t4. Delete at Beginning\n");
  node *temp = head;
                                                           printf("\t\t5. Delete at Position\n");
                                                           printf("\t\t6. Delete at End\n");
  for (int i = 1; i < pos \&\& temp != NULL;
i++)
                                                           printf("\t\t7. Exit\n\n");
                                                           printf("\tEnter your choice: ");
     temp = temp -> next;
                                                           scanf("%d", &choice);
  if (temp != NULL) {
     if (temp->prev != NULL)
                                                           switch (choice) {
       temp->prev->next = temp->next;
                                                             case 1:
     if (temp->next != NULL)
                                                                printf("\n\tEnter value to insert: ");
       temp->next->prev=temp->prev;
                                                                scanf("%d", &value);
  } else {
                                                                insertAtBeg(value);
    printf("\n\t\t!!! Invalid Position...\n");
                                                                break;
                                                             case 2:
                                                                printf("\n\tEnter position: ");
                                                                scanf("%d", &pos);
```

```
printf("\tEnter value to insert: ");
                                                         deleteAtEnd();
  scanf("%d", &value);
                                                         break:
  insertAtPos(pos, value);
                                                      case 7:
  break:
                                                         printf("\n\tExiting\ program...\n");
case 3:
  printf("\n\tEnter value to insert: ");
                                                      default:
  scanf("%d", &value);
                                                         printf("\n\tInvalid choice! Please try
  insertAtEnd(value);
                                               again.\n'');
  break:
                                                    printf("\n\tPress Enter to continue...");
case 4:
  deleteAtBeg();
                                                    getch();
                                                 } while (choice != 7);
  break;
case 5:
  printf("\n\tEnter position: ");
  scanf("%d", &pos);
                                               int main() {
  deleteAtPos(pos);
                                                 Dashboard();
  break:
                                                 return 0;
case 6:
```

```
C:\Users\Mahesh\Desktop\Linklist in C\output\DoublyLinkedList.exe

List of data:
65 <-> 19 <-> 51 <-> 17 <-> NULL

Enter operation:

1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Delete at Beginning
5. Delete at Position
6. Delete at End
7. Exit

Enter your choice:
```

```
List of data:
65 <-> 19 <-> 51 <-> 17 <-> NULL

Enter operation:

1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Delete at Beginning
5. Delete at Position
6. Delete at End
7. Exit

Enter your choice: 2

Enter position: 3
Enter value to insert: 92

Press Enter to continue...

C:\Users\Mahesh\Desktop\Linklist in C\output\DoublyLinkedList.ex
```

C:\Users\Mahesh\Desktop\Linklist in C\output\DoublyLinkedList.exe

```
C:\Users\Mahesh\Desktop\Linklist in C\output\DoublyLinkedList.exe

List of data:
65 <-> 19 <-> 92 <-> 51 <-> 17 <-> NULL

Enter operation:

1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Delete at Beginning
5. Delete at Position
6. Delete at End
7. Exit

Enter your choice: 4

Press Enter to continue...
```

```
C:\Users\Mahesh\Desktop\Linklist in C\output\DoublyLinkedList.ex

List of data:

19 <-> 92 <-> 51 <-> 17 <-> NULL

Enter operation:

1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Delete at Beginning
5. Delete at Position
6. Delete at End
7. Exit

Enter your choice: 7

Exiting program...

Press Enter to continue...
```

RESULTS AND DISCUSSION:

The stuents are successful to write code for doubly linked list in C programming. This program helps to understand basics of Data structure. The program have menu driven functioality to perform operations in doubly linked list.

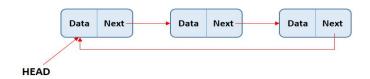
CONCLUSION:

This laboratory exercise provided a hands-on experience in DSA. Students gained practical knowledge of implementing basic in doubly linked list and now better equipped to undertake more complex programming tasks in the future.

OBJECTIVE: To implement Circular Singly Linked List and perform various operations in C programming.

THEORY:

A "circular singly linked list" is a type of linked list where each node only points to the next node in the sequence, but the last node in the list points back to the first node, creating a continuous loop or "circle" allowing for traversal from any point in the list back to the beginning without reaching a null pointer; essentially, it's a singly linked list where the last node connects to the first node.



Operations to perform in circular singly linked list are mention bellow:

- 1. Insert Operation
 - a) Insert at beginning
 - b) Insert at Position
 - c) Insert at End

- 2. Delete Operation
 - a) Delete at beginning
 - b) Delete at position
 - c) Delere at End
- 3. Traverse Operation
 - a) Display items

PROGRAMS

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
// Structure for Singly Circular Linked List
Node
struct Node {
   int data:
   struct Node *next:
};
typedef struct Node node;
node *head = NULL;
// Function to create a new node
node* createNewNode(int item) {
   node *newnode =
(node*)malloc(sizeof(node));
   newnode -> data = item;
   newnode->next = newnode; // Circular
connection
   return newnode;
// Function to insert at the beginning
```

```
void insertAtBeg(int item) {
   node *newnode = createNewNode(item);
   if (head == NULL)  {
       head = newnode;
   } else {
       node *temp = head;
       while (temp->next != head)
           temp = temp -> next;
       temp->next = newnode;
       newnode -> next = head;
       head = newnode;
// Function to insert at a specific position
void insertAtPos(int pos, int item) {
   if (pos < 1) {
       printf("\n\t\t!!! Invalid Position...\n");
       return:
   if (pos == 1 || head == NULL) {
       insertAtBeg(item);
       return;
```

```
node *temp = head;
                                                         node *temp = head;
   node *newnode = createNewNode(item);
                                                        for (int i = 1; i < pos-1; i++)
   for (int i = 1; i < pos - 1; i++)
                                                             temp = temp -> next;
       temp = temp -> next;
                                                         temp->next = (temp->next)->next;
   newnode->next = temp->next;
   temp->next = newnode;
                                                     // Function to delete at the end
                                                     void deleteAtEnd() {
// Function to insert at the end
                                                         if (head == NULL)  {
void insertAtEnd(int item) {
                                                            printf("\n\t!!! Empty Node list...\n");
   node *newnode = createNewNode(item);
                                                            return;
   if (head == NULL)  {
       head = newnode;
                                                         if (head - next = head) { // Only one node}
   } else {
                                                            free(head);
       node *temp = head;
                                                            head = NULL;
       while (temp->next != head)
                                                         } else {
           temp = temp -> next;
                                                            node *temp = head;
       temp->next = newnode;
                                                            while ((temp->next)->next != head)
                                                                temp = temp -> next;
       newnode -> next = head;
                                                            free(temp->next);
                                                            temp->next = head;
// Function to delete at the beginning
void deleteAtBeg() {
   if (head == NULL) \{
                                                     // Function to display the circular linked list
       printf("\n\t!!! Empty Node list...\n");
                                                     void display() {
                                                         printf("\n\tList\ of\ data:\n\t");
   if (head->next == head) head = NULL;
                                                         if (head == NULL)  {
                                                            printf("\t!!! Empty Node list...\n");
   else {
       node *temp = head;
                                                            return;
       while (temp->next != head)
           temp = temp -> next;
                                                         node *temp = head;
       head = head -> next;
                                                         do {
       temp->next = head;
                                                            printf("\%d -> ", temp->data);
                                                            temp = temp -> next;
                                                         } while (temp != head);
// Function to delete at a specific position
                                                         printf("(HEAD)\n");
void deleteAtPos(int pos) {
   if (head == NULL) \{
       printf("\n\t!!! Empty Node list...\n");
                                                     // Menu-driven Dashboard
                                                     void Dashboard() {
       return;
                                                         int choice, value, pos;
   if (pos < 1) {
                                                         do {
       printf("\n\t\!!! Invalid Position...\n");
                                                            system("cls");
       return;
                                                            display();
   if (pos == 1) {
                                                            printf("\n\tEnter operation: \n");
                                                            printf("\t\t1. Insert at Beginning\n");
       deleteAtBeg();
                                                            printf("\t\t2. Insert at Position\n");
       return;
                                                            printf("\t \t 3. Insert at End\n");
                                                            printf("\t\t4. Delete at Beginning\n");
```

```
printf("\t\t5. Delete at Position\n");
                                                                     break:
       printf("\t \ believe at End\n");
                                                                 case 5:
       printf("\t\t7. Exit\n\n");
                                                                     printf("\n\tEnter position: ");
       printf("\tEnter your choice: ");
                                                                     scanf("%d", &pos);
       scanf("%d", &choice);
                                                                     deleteAtPos(pos);
                                                                     break:
       switch (choice) {
                                                                  case 6:
                                                                     deleteAtEnd();
           case 1:
               printf("\n\tEnter value to insert:
                                                                     break;
");
                                                                  case 7:
               scanf("%d", &value);
                                                                     printf("\n\tExiting\ program...\n");
               insertAtBeg(value);
                                                                     break;
                                                                 default:
               break;
                                                                     printf("\n\tInvalid choice! Please
           case 2:
                                                      try again. n'');
               printf("\n\tEnter position: ");
               scanf("%d", &pos);
               printf("\tEnter value to insert: ");
                                                              printf("\n\tPress Enter to continue...");
               scanf("%d", &value);
                                                              getch();
               insertAtPos(pos, value);
                                                          } while (choice != 7);
               break;
           case 3:
               printf("\n\tEnter value to insert:
                                                      int main() {
");
                                                          Dashboard();
               scanf("%d", &value);
                                                          return 0;
               insertAtEnd(value);
               break;
           case 4:
               deleteAtBeg();
```

```
C:\Users\Mahesh\Desktop\Linklist in C\output\CircularSinglyLinkedList.ex

List of data:
33 -> 65 -> 18 -> 52 -> (HEAD)

Enter operation:

1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Delete at Beginning
5. Delete at Position
6. Delete at End
7. Exit

Enter your choice: 2

Enter position: 5
Enter value to insert: 47

Press Enter to continue...
```

```
C:\Users\Mahesh\Desktop\Linklist in C\output\SingleLinkedList.exe

List of data:

15 -> 20 -> 5 -> 10 -> 100 -> 50 -> 60 -> NULL

Enter operation:

1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Delete at Beginning
5. Delete at Position
6. Delete at End
7. Exit

Enter your choice: 1

Enter value to insert: 73

Press Enter to continue..._
```

```
C:\Users\Mahesh\Desktop\Linklist in C\output\CircularSinglyLinkedList.exe
        List of data:
        76 -> 44 -> 65 -> 23 -> 80 -> (HEAD)
        Enter operation:
                1. Insert at Beginning
                2. Insert at Position
                3. Insert at End
                4. Delete at Beginning
                5. Delete at Position
                6. Delete at End
                7. Exit
        Enter your choice: 1
        Enter value to insert: 26
        Press Enter to continue..._
C:\Users\Mahesh\Desktop\Linklist in C\output\CircularSinglyLinkedList.exe
        List of data:
        26 -> 76 -> 44 -> 65 -> 23 -> 80 -> (HEAD)
        Enter operation:
                1. Insert at Beginning
                2. Insert at Position
                3. Insert at End
                4. Delete at Beginning
                5. Delete at Position
                6. Delete at End
                7. Exit
        Enter your choice: 7
        Exiting program...
        Press Enter to continue..._
```

RESULTS AND DISCUSSION:

The stuents are successful to write code for circular single linked list in C programming. This program helps to understand basics of Data structure to create CSLL. The program have menu driven functionality to perform operations in circular singly linked list.

CONCLUSION:

This laboratory exercise provided a hands-on experience in DSA. Students gained practical knowledge of implementing basic in circular singly linked list and now better equipped to undertake more complex programming tasks in the future.

OBJECTIVE: To implement Circular Doubly Linked List and perform various operations in C programming.

THEORY:

A "circular doubly linked list" is a type of linked list where each node points to both its previous and next nodes, and the last node in the list connects back to the first node, creating a circular structure that allows for bidirectional traversal through the list.



Operations to perform in circular doubly linked list are mention bellow:

- 1. Insert Operation
 - a) Insert at beginning
 - b) Insert at Position
 - c) Insert at End

- 2. Delete Operation
 - a) Delete at beginning
 - b) Delete at position
 - c) Delere at End
- 3. Traverse Operation
 - a) Display items

PROGRAMS

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
// Structure for Circular Doubly Linked List
struct Node {
  int data:
  struct Node *next:
  struct Node *prev;
};
typedef struct Node node;
node *head = NULL:
// Function to create a new node
node* createNewNode(int item) {
  node *newnode =
(node*)malloc(sizeof(node));
  newnode -> data = item;
  newnode -> next = newnode:
  newnode \rightarrow prev = newnode;
  return newnode;
// Function to insert at the beginning
void insertAtBeg(int item) {
  node *newnode = createNewNode(item);
  if (head == NULL)  {
     head = newnode;
  } else {
```

```
node *tail = head->prev; // Last node
    newnode -> next = head;
    newnode -> prev = tail;
    tail->next = newnode;
    head->prev = newnode;
    head = newnode; // Update head
// Function to insert at a specific position
void insertAtPos(int pos, int item) {
  if (pos < 1) {
    printf("\n\t\t!!! Invalid Position...\n");
    return;
  if(pos == 1 \mid\mid head == NULL) {
    insertAtBeg(item);
    return;
  node *temp = head;
  node *newnode = createNewNode(item);
  for (int i = 1; i < pos - 1; i++)
    temp = temp -> next;
  newnode->next = temp->next;
  newnode -> prev = temp;
  temp->next->prev=newnode;
  temp->next = newnode;
```

```
temp->next = delNode->next;
                                                        delNode->next->prev=temp;
// Function to insert at the end
                                                       free(delNode);
void insertAtEnd(int item) {
  node *newnode = createNewNode(item);
  if (head == NULL)  {
                                                     // Function to delete at the end
     head = newnode;
                                                     void deleteAtEnd() {
                                                        if (head == NULL) {
  } else {
     node *tail = head->prev; // Last node
                                                          printf("\n\t!!! Empty Node list...\n");
     tail->next = newnode;
                                                          return;
     newnode -> prev = tail;
     newnode -> next = head;
                                                        if (head - next = head) { // Only one node}
     head->prev = newnode;
                                                          free(head);
                                                          head = NULL;
                                                       } else {
                                                          node *tail = head -> prev;
                                                          node *newTail = tail->prev;
// Function to delete at the beginning
void deleteAtBeg() {
                                                          newTail->next = head;
  if (head == NULL)  {
                                                          head->prev = newTail;
                                                          free(tail);
    printf("\n\t!!! Empty Node list...\n");
  if (head - next = head) { // Only one node}
                                                     // Function to display the circular doubly linked
    free(head);
    head = NULL;
                                                     list
  } else {
                                                     void display() {
     node *tail = head -> prev;
                                                       printf("\n\tList\ of\ data:\n\t");
     node *delNode = head;
     head = head -> next;
                                                        if (head == NULL)  {
     head->prev = tail;
                                                          printf("\t!!! Empty Node list...\n");
     tail->next = head;
                                                          return;
    free(delNode);
                                                        node *temp = head;
                                                        do {
                                                          printf("\%d <->", temp->data);
// Function to delete at a specific position
void deleteAtPos(int pos) {
                                                          temp = temp -> next;
                                                        } while (temp != head);
  if (head == NULL) \{
    printf("\n\t!!! Empty Node list...\n");
                                                       printf("(HEAD)\n");
     return;
  if (pos < 1) {
    printf("\n\t\t!!! Invalid Position...\n");
                                                     // Menu-driven DSAhboard
                                                     void DSAhboard() {
     return;
                                                        int choice, value, pos;
  if (pos == 1) {
                                                        do {
     deleteAtBeg();
                                                          system("cls");
     return;
                                                          display();
                                                          printf("\n\tEnter operation: \n");
                                                          printf("\t\t1. Insert at Beginning\n");
  node *temp = head;
                                                          printf("\t\t2. Insert at Position\n");
                                                          printf("\t\t3. Insert at End\n");
  for (int i = 1; i < pos - 1; i++)
                                                          printf("\t\t4. Delete at Beginning\n");
     temp = temp -> next;
                                                          printf("\t\t5. Delete at Position\n");
                                                          printf("\t \ Delete \ at \ End\n");
  node *delNode = temp->next;
```

```
printf("\t\t7. Exit\n\n");
                                                            printf("\n\tEnter position: ");
printf("\tEnter your choice: ");
                                                            scanf("%d", &pos);
scanf("%d", &choice);
                                                            deleteAtPos(pos);
                                                            break:
switch (choice) {
                                                          case 6:
  case 1:
                                                            deleteAtEnd();
     printf("\n\tEnter value to insert: ");
                                                            break;
     scanf("%d", &value);
                                                          case 7:
     insertAtBeg(value);
                                                            printf("\n\tExiting\ program...\n");
     break;
                                                            break;
  case 2:
                                                          default:
     printf("\n\tEnter position: ");
                                                            printf("\n\tInvalid choice! Please try
     scanf("%d", &pos);
                                                  again. \langle n'' \rangle;
     printf("\tEnter value to insert: ");
                                                       }
     scanf("%d", &value);
                                                       printf("\n\tPress Enter to continue...");
     insertAtPos(pos, value);
                                                       getch();
                                                     } while (choice != 7);
     break:
  case 3:
     printf("\n\tEnter value to insert: ");
     scanf("%d", &value);
                                                  int main() {
     insertAtEnd(value);
                                                     DSAhboard();
     break;
                                                     return 0;
  case 4:
     deleteAtBeg();
     break:
  case 5:
```

```
C:\Users\Mahesh\Desktop\Linklist in C\output\CircularDoublyLinkedLinklist of data:

18 <-> 53 <-> 15 <-> 46 <-> (HEAD)

Enter operation:

1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Delete at Beginning
5. Delete at Position
6. Delete at End
7. Exit

Enter your choice: 5

Enter position: 3

Press Enter to continue...
```

```
\Users\Mahesh\Desktop\Linklist in C\output\CircularDoublyLink

List of data:

18 <-> 53 <-> 46 <-> (HEAD)

Enter operation:

1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Delete at Beginning
5. Delete at Position
6. Delete at End
7. Exit

Enter your choice: 6

Press Enter to continue...
```

```
\Users\Mahesh\Desktop\Linklist in C\output\CircularDoublyLinke
List of data:
   18 <-> 53 <-> (HEAD)

Enter operation:
        1. Insert at Beginning
        2. Insert at Position
        3. Insert at End
        4. Delete at Beginning
        5. Delete at Position
        6. Delete at End
        7. Exit

Enter your choice: 7

Exiting program...

Press Enter to continue...
```

RESULTS AND DISCUSSION:

The stuents are successful to write code for circular doubly linked list in C programming. This program helps to understand basics of Data structure to create CDLL. The program have menu driven functionality to perform operations in circular doubly linked list.

CONCLUSION:

This laboratory exercise provided a hands-on experience in DSA. Students gained practical knowledge of implementing basic in circular doubly linked list and now better equipped to undertake more complex programming tasks in the future.