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

2nd Year / 3rd Semester

INDEX

Name: Mahesh Kumar Udas — Roll No.: 21

Faculty: BSc.CSIT — Semester: Third

Subject: Data Structure and Algorithm — Year : Second

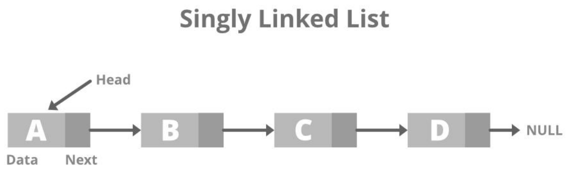
S.N.	**Title**	**Date of Submission**	**Sign**

LAB 1

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



.

Operations to perform in single linked list are mention bellow:

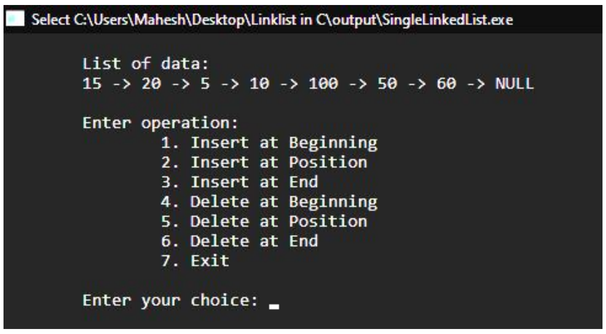
1. Insert Operation
   a) Insert at beginning
   b) Insert at Position
   c) Insert at End
2. Delete Operation
   a) Delete at beginning
   b) Delete at position
   c) Delere at End
3. Traverse Operation
   a) Display items

PROGRAMS

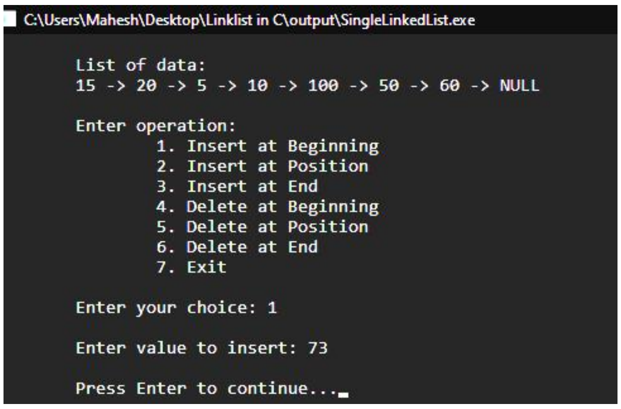
```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>

// 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\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) head = newnode;

	node *temp = head;
	while (temp->next != NULL)
		temp = temp->next;
	temp->next = newnode;
}

// Function to delete at the beginning
void deleteAtBeg(){
	if(head==NULL) printf("\n\t!!!Empty
Node list...\n");
	else head = head->next;
}

// Function to delete at position
void deleteAtPos(int pos){
	if (pos < 1) {
		printf("\n\t\t!!! Invalid Position...\n");
		return;
	}
	if (pos == 1) {
		deleteAtBeg();
		return;
	}

	node *temp = head;

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

	if (temp != NULL)
		temp->next = (temp->next)->next;
	else {
		printf("\n\t\t!!! Invalid Position...\n");
	}
}

// Function to delete at end
void deleteAtEnd(){
	if(head==NULL) printf("\n\t!!!Empty
Node list...\n");
	else{
		node *temp = head;
		while((temp->next)->next!=NULL)
			temp = temp->next;
		temp->next = NULL;
	}
}

// Function to display the linked list
void display() {
	node *temp = head;
	printf("\n\tList of data:\n\t");
	if (head == NULL) {
		printf("\t!!! Empty Node list...\n");
		return;
	}
	while (temp != NULL) {
		printf("%d -> ", temp->data);
		temp = temp->next;
	}
	printf("NULL\n");
}

// Menu-driven Dashboard
void Dashboard() {
	int choice, value, pos;
	do {
		system("cls");

		display();
		printf("\n\tEnter operation: \n");
		printf("\t\t1. Insert at Beginning\n");
		printf("\t\t2. Insert at Position\n");
		printf("\t\t3. Insert at End\n");
		printf("\t\t4. Delete at Beginning\n");
		printf("\t\t5. Delete at Position\n");
		printf("\t\t6. Delete at End\n");
		printf("\t\t7. Exit\n\n");
		printf("\tEnter your choice: ");
		scanf("%d", &choice);

		switch (choice) {
			case 1:
				printf("\n\tEnter value to
insert: ");
				scanf("%d", &value);
				insertAtBeg(value);
				break;
			case 2:
				printf("\n\tEnter position: ");
				scanf("%d", &pos);
				printf("\tEnter value to
insert: ");
				scanf("%d", &value);
				insertAtPos(pos, value);
				break;
			case 3:
				printf("\n\tEnter value to
insert: ");
				scanf("%d", &value);
				insertAtEnd(value);
				break;
			case 4:
				deleteAtBeg();
				break;
			case 5:
				printf("\n\tEnter position: ");
				scanf("%d", &pos);
```

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

int main() {
     Dashboard();
    return 0;
}
```

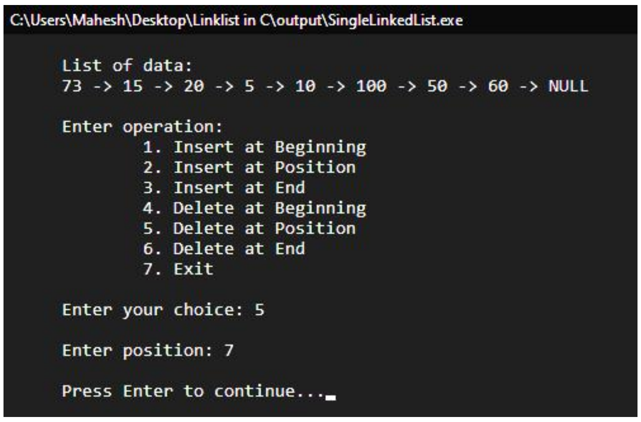
Output:



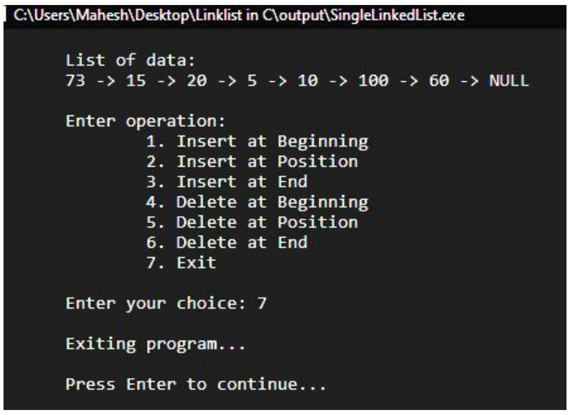
```
Select C:\Users\Mahesh\Desktop\Linklist in C\output\SingleLinkedList.exe

     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 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 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:

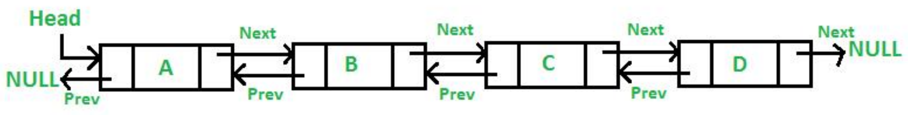
This laboratory exercise provided a hands-on experience in DSA. Students gained practical knowledge of implementing basic in single linked list and now better equipped to undertake more complex programming tasks in the future.

LAB 2

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:

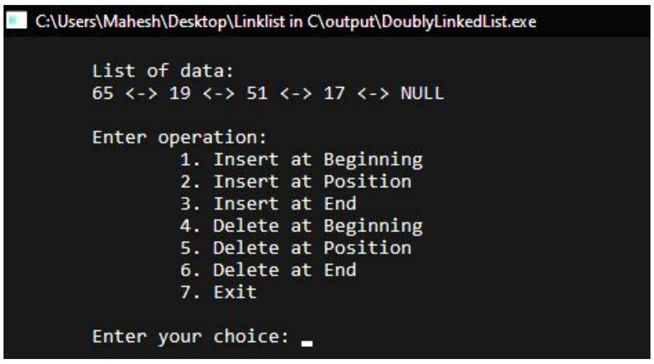
1. Insert Operation
   a) Insert at beginning
   b) Insert at Position
   c) Insert at End
2. Delete Operation
   a) Delete at beginning
   b) Delete at position
   c) Delere at End
3. Traverse Operation
   a) Display items

PROGRAM:

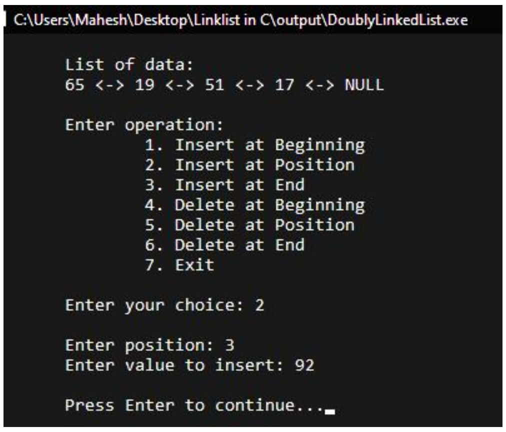
```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>

// 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\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;
```

```
        newnode->prev = temp;
    } 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) {
        head = newnode;
        return;
    }
    node *temp = head;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newnode;
    newnode->prev = temp;
}

// Function to delete at the beginning
void deleteAtBeg(){
    if(head == NULL) printf("\n\t!!!Empty Node
list...\n");
    else {
        head = head->next;
        if (head != NULL)
            head->prev = NULL;
    }
}

// Function to delete at position
void deleteAtPos(int pos){
    if (pos < 1) {
        printf("\n\t\t!!! Invalid Position...\n");
        return;
    }
    if (pos == 1) {
        deleteAtBeg();
        return;
    }

    node *temp = head;
    for (int i = 1; i < pos && temp != NULL;
i++)
        temp = temp->next;

    if (temp != NULL) {
        if (temp->prev != NULL)
            temp->prev->next = temp->next;
        if (temp->next != NULL)
            temp->next->prev = temp->prev;
    } else {
        printf("\n\t\t!!! Invalid Position...\n");
    }
}

// Function to delete at end
void deleteAtEnd(){
    if(head == NULL) printf("\n\t!!!Empty Node
list...\n");
    else {
        node *temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        if (temp->prev != NULL)
            temp->prev->next = NULL;
        else
            head = NULL;
    }
}

// Function to display the linked list
void display() {
    node *temp = head;
    printf("\n\tList of data:\n\t");
    if (head == NULL) {
        printf("\t!!! Empty Node list...\n");
        return;
    }
    while (temp != NULL) {
        printf("%d <-> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

// Menu-driven Dashboard
void Dashboard() {
    int choice, value, pos;
    do {
        system("cls");

        display();
        printf("\n\tEnter operation: \n");
        printf("\t\t1. Insert at Beginning\n");
        printf("\t\t2. Insert at Position\n");
        printf("\t\t3. Insert at End\n");
        printf("\t\t4. Delete at Beginning\n");
        printf("\t\t5. Delete at Position\n");
        printf("\t\t6. Delete at End\n");
        printf("\t\t7. Exit\n\n");
        printf("\tEnter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("\n\tEnter value to insert: ");
                scanf("%d", &value);
                insertAtBeg(value);
                break;
            case 2:
                printf("\n\tEnter position: ");
                scanf("%d", &pos);
            printf("\tEnter value to insert: ");
            scanf("%d", &value);
            insertAtPos(pos, value);
            break;
        case 3:
            printf("\n\tEnter value to insert: ");
            scanf("%d", &value);
            insertAtEnd(value);
            break;
        case 4:
            deleteAtBeg();
            break;
        case 5:
            printf("\n\tEnter position: ");
            scanf("%d", &pos);
            deleteAtPos(pos);
            break;
        case 6:
            deleteAtEnd();
            break;
        case 7:
            printf("\n\tExiting program...\n");
            break;
        default:
            printf("\n\tInvalid choice! Please try
again.\n");
        }
        printf("\n\tPress Enter to continue...");
        getch();
    } while (choice != 7);
}

int main() {
    Dashboard();
    return 0;
}
```

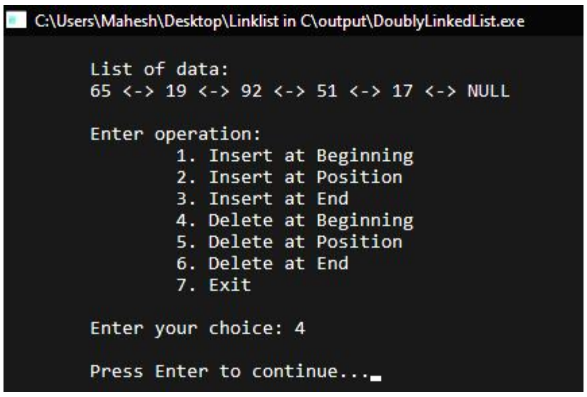
Output:



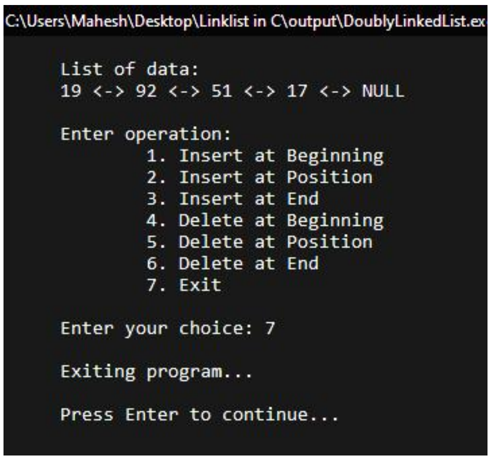
```
C:\Users\Mahesh\Desktop\Linklist in C\output\DoublyLinkedList.exe

        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: _
```



```
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: 2

        Enter position: 3
        Enter value to insert: 92

        Press Enter to continue..._
```



```
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:

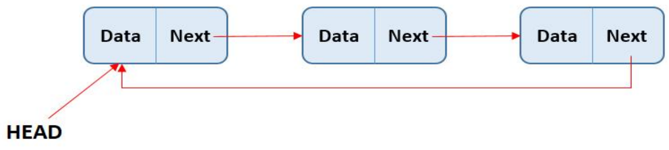
This laboratory exercise provided a hands-on experience in DSA. Students gained practical knowledge of implementing basic in doubly linked list and now better equipped to undertake more complex programming tasks in the future.

LAB 3

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.



.

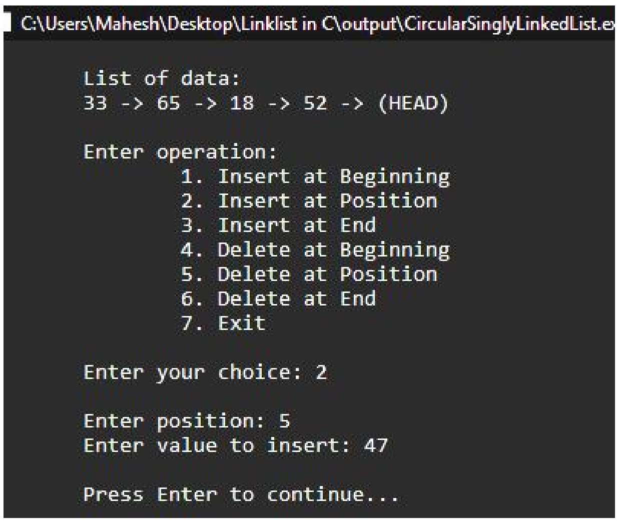
Operations to perform in circular singly linked list are mention bellow:

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

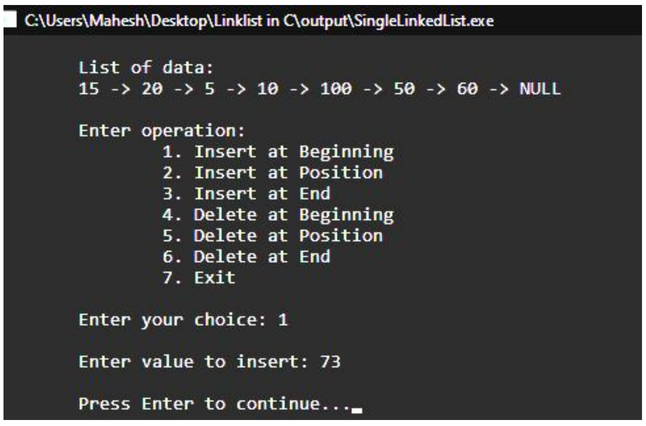
PROGRAMS

```
#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 *newnode = createNewNode(item);

    for (int i = 1; i < pos - 1; i++)
        temp = temp->next;

    newnode->next = temp->next;
    temp->next = newnode;
}

// Function to insert at the end
void insertAtEnd(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;
    }
}

// Function to delete at the beginning
void deleteAtBeg() {
    if (head == NULL) {
        printf("\n\t!!! Empty Node list...\n");
        return;
    }
    if (head->next == head) head = NULL;
    else {
        node *temp = head;
        while (temp->next != head)
            temp = temp->next;

        head = head->next;
        temp->next = head;
    }
}

// Function to delete at a specific position
void deleteAtPos(int pos) {
    if (head == NULL) {
        printf("\n\t!!! Empty Node list...\n");
        return;
    }
    if (pos < 1) {
        printf("\n\t\t!!! Invalid Position...\n");
        return;
    }
    if (pos == 1) {
        deleteAtBeg();
        return;
    }
    node *temp = head;

    for (int i = 1; i < pos-1; i++)
        temp = temp->next;

    temp->next = (temp->next)->next;
}

// Function to delete at the end
void deleteAtEnd() {
    if (head == NULL) {
        printf("\n\t!!! Empty Node list...\n");
        return;
    }
    if (head->next == head) { // Only one node
        free(head);
        head = NULL;
    } else {
        node *temp = head;
        while ((temp->next)->next != head)
            temp = temp->next;
        free(temp->next);
        temp->next = head;
    }
}

// Function to display the circular linked list
void display() {
    printf("\n\tList of data:\n\t");

    if (head == NULL) {
        printf("\t!!! Empty Node list...\n");
        return;
    }

    node *temp = head;
    do {
        printf("%d -> ", temp->data);
        temp = temp->next;
    } while (temp != head);

    printf("(HEAD)\n");
}

// Menu-driven Dashboard
void Dashboard() {
    int choice, value, pos;
    do {
        system("cls");

        display();
        printf("\n\tEnter operation: \n");
        printf("\t\t1. Insert at Beginning\n");
        printf("\t\t2. Insert at Position\n");
        printf("\t\t3. Insert at End\n");
        printf("\t\t4. Delete at Beginning\n");
        printf("\t\t5. Delete at Position\n");
        printf("\t\t6. Delete at End\n");
        printf("\t\t7. Exit\n\n");
        printf("\tEnter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("\n\tEnter value to insert:
");
                scanf("%d", &value);
                insertAtBeg(value);
                break;
            case 2:
                printf("\n\tEnter position: ");
                scanf("%d", &pos);
                printf("\tEnter value to insert: ");
                scanf("%d", &value);
                insertAtPos(pos, value);
                break;
            case 3:
                printf("\n\tEnter value to insert:
");
                scanf("%d", &value);
                insertAtEnd(value);
                break;
            case 4:
                deleteAtBeg();
                break;
            case 5:
                printf("\n\tEnter position: ");
                scanf("%d", &pos);
                deleteAtPos(pos);
                break;
            case 6:
                deleteAtEnd();
                break;
            case 7:
                printf("\n\tExiting program...\n");
                break;
            default:
                printf("\n\tInvalid choice! Please
try again.\n");
        }
        printf("\n\tPress Enter to continue...");
        getch();
    } while (choice != 7);
}

int main() {
    Dashboard();
    return 0;
}
```

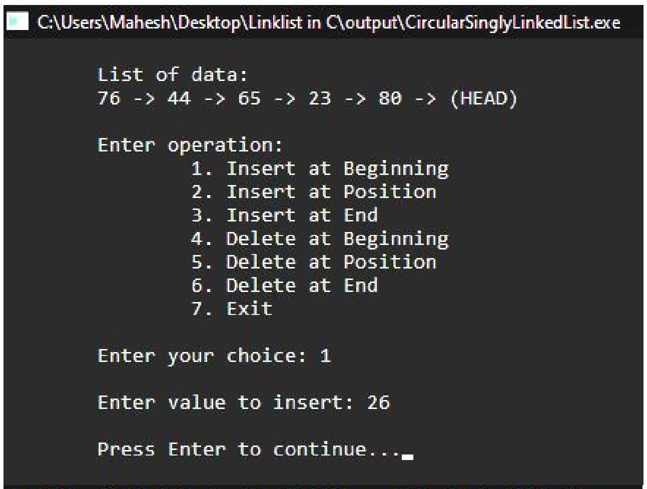
Output:



```
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...
```

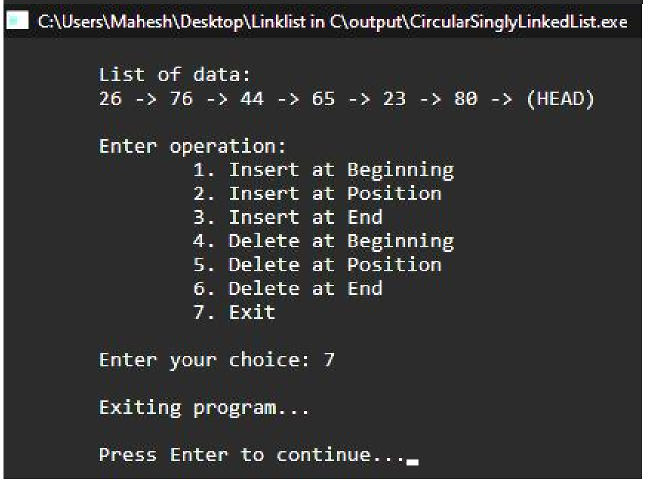


```
C:\Users\Mahesh\Desktop\Linklist in C\output\SingleLinkedList.exe

        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 functioality to perform operations in circular singly linked list.

CONCLUSION:

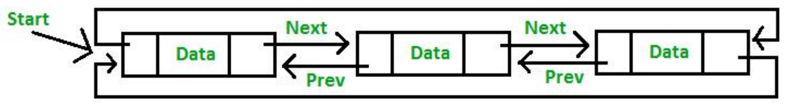
This laboratory exercise provided a hands-on experience in DSA. Students gained practical knowledge of implementing basic in circular singly linked list and now better equipped to undertake more complex programming tasks in the future.

LAB 4

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

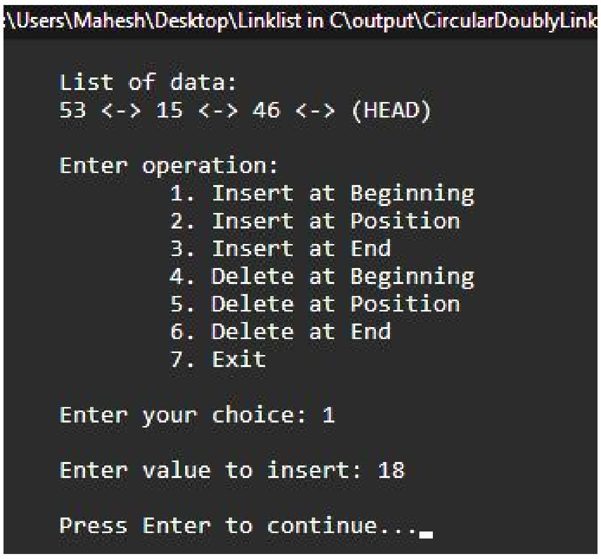
PROGRAMS

```
#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->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 || 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;
```

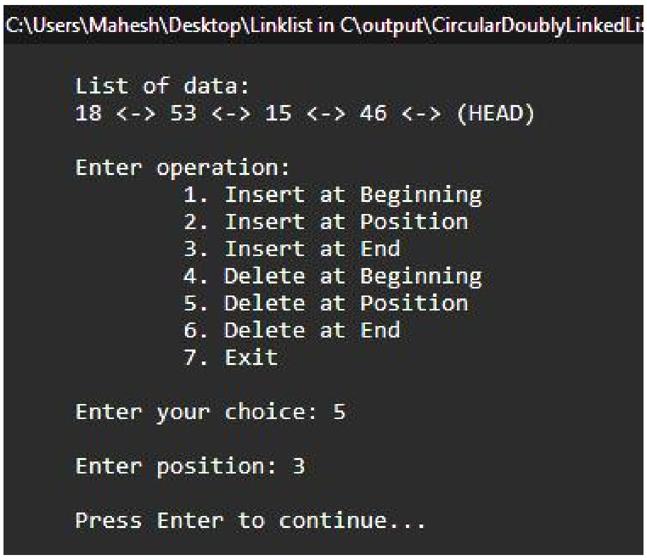
```
}

// Function to insert at the end
void insertAtEnd(int item) {
   node *newnode = createNewNode(item);
   if (head == NULL) {
      head = newnode;
   } else {
      node *tail = head->prev; // Last node
      tail->next = newnode;
      newnode->prev = tail;
      newnode->next = head;
      head->prev = newnode;
   }
}

// Function to delete at the beginning
void deleteAtBeg() {
   if (head == NULL) {
      printf("\n\t!!! Empty Node list...\n");
      return;
   }
   if (head->next == head) { // Only one node
      free(head);
      head = NULL;
   } else {
      node *tail = head->prev;
      node *delNode = head;
      head = head->next;
      head->prev = tail;
      tail->next = head;
      free(delNode);
   }
}

// Function to delete at a specific position
void deleteAtPos(int pos) {
   if (head == NULL) {
      printf("\n\t!!! Empty Node list...\n");
      return;
   }
   if (pos < 1) {
      printf("\n\t\t!!! Invalid Position...\n");
      return;
   }
   if (pos == 1) {
      deleteAtBeg();
      return;
   }

   node *temp = head;

   for (int i = 1; i < pos - 1; i++)
      temp = temp->next;

   node *delNode = temp->next;
   temp->next = delNode->next;
   delNode->next->prev = temp;
   free(delNode);
}

// Function to delete at the end
void deleteAtEnd() {
   if (head == NULL) {
      printf("\n\t!!! Empty Node list...\n");
      return;
   }
   if (head->next == head) { // Only one node
      free(head);
      head = NULL;
   } else {
      node *tail = head->prev;
      node *newTail = tail->prev;
      newTail->next = head;
      head->prev = newTail;
      free(tail);
   }
}

// Function to display the circular doubly linked
list
void display() {
   printf("\n\tList of data:\n\t");

   if (head == NULL) {
      printf("\t!!! Empty Node list...\n");
      return;
   }

   node *temp = head;
   do {
      printf("%d <-> ", temp->data);
      temp = temp->next;
   } while (temp != head);

   printf("(HEAD)\n");
}

// Menu-driven DSAhboard
void DSAhboard() {
   int choice, value, pos;
   do {
      system("cls");

      display();
      printf("\n\tEnter operation: \n");
      printf("\t\t1. Insert at Beginning\n");
      printf("\t\t2. Insert at Position\n");
      printf("\t\t3. Insert at End\n");
      printf("\t\t4. Delete at Beginning\n");
      printf("\t\t5. Delete at Position\n");
      printf("\t\t6. Delete at End\n");
```

```
        printf("\t\t7. Exit\n\n");
        printf("\tEnter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("\n\tEnter value to insert: ");
                scanf("%d", &value);
                insertAtBeg(value);
                break;
            case 2:
                printf("\n\tEnter position: ");
                scanf("%d", &pos);
                printf("\tEnter value to insert: ");
                scanf("%d", &value);
                insertAtPos(pos, value);
                break;
            case 3:
                printf("\n\tEnter value to insert: ");
                scanf("%d", &value);
                insertAtEnd(value);
                break;
            case 4:
                deleteAtBeg();
                break;
            case 5:
                printf("\n\tEnter position: ");
                scanf("%d", &pos);
                deleteAtPos(pos);
                break;
            case 6:
                deleteAtEnd();
                break;
            case 7:
                printf("\n\tExiting program...\n");
                break;
            default:
                printf("\n\tInvalid choice! Please try
again.\n");
        }
        printf("\n\tPress Enter to continue...");
        getch();
    } while (choice != 7);
}

int main() {
    DSAhboard();
    return 0;
}
```

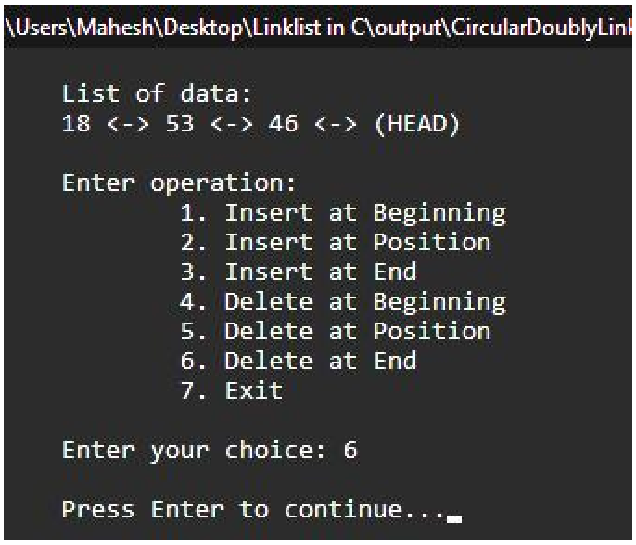
Output:



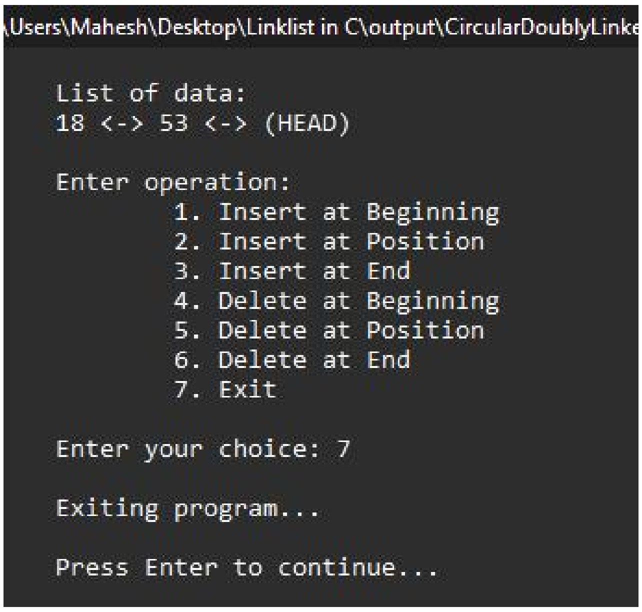
```
:\Users\Mahesh\Desktop\Linklist in C\output\CircularDoublyLink

    List of data:
    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: 1

    Enter value to insert: 18

    Press Enter to continue...
```



```
C:\Users\Mahesh\Desktop\Linklist in C\output\CircularDoublyLinkedLi

    List 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 functioality 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.