**Practical – 4**

AIM: Implementation and time analysis of Circular linked list .

#include <iostream>

using namespace std;

class Node

{

public:

int data;

Node \*next;

Node(int x)

{

this->data = x;

this->next = NULL;

};

};

void InsertAtStart(Node \*&head, int value)

{

Node \*newNode = new Node(value);

if (head == NULL)

{

head = newNode;

head->next = head;

return;

}

Node \*temp = head;

while (temp->next != head)

{

temp = temp->next;

}

newNode->next = temp->next;

temp->next = newNode;

head = newNode;

}

void InsertAtlast(Node \*head, int value)

{

Node \*newNode = new Node(value);

if (head == NULL)

{

head = newNode;

head->data = newNode->data;

head->next = head;

return;

}

Node \*temp = head;

while (temp->next != head)

{

temp = temp->next;

}

newNode->next = temp->next;

temp->next = newNode;

}

void insertInOrder(Node \*&head, int value)

{

bool value\_inserted = false;

Node \*newNode = new Node(value);

if (head == NULL)

{

head = newNode;

head->next = head;

return;

}

Node \*temp = head;

if (value <= head->data)

{

Node \*temp = head;

while (temp->next != head)

{

temp = temp->next;

}

newNode->next = temp->next;

temp->next = newNode;

head = newNode;

value\_inserted = true;

}

else

{

while (temp->next != head && temp->next->data < value)

{

temp = temp->next;

}

newNode->next = temp->next;

temp->next = newNode;

}

}

void deleteNodeByValue(Node \*&head, int value)

{

if (head == NULL)

{

cout << "Linked list is empty";

return;

}

Node \*temp = head;

while (temp->next != head && temp->next->data != value)

{

temp = temp->next;

}

if (temp->next == temp)

{

head->next == NULL;

head->data = 0;

return;

}

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

}

void display(Node \*head)

{

if (head == NULL)

{

cout << "linked list is empty" << endl;

return;

}

Node \*temp = head;

int counter = 0;

do

{

cout << temp->data << "->";

temp = temp->next;

counter++;

} while (temp != head);

cout << endl;

cout << "|";

for (int i = 0; i <= counter; i++)

{

cout << " ";

}

cout << " |";

cout << endl;

for (int i = 0; i <= counter + 1; i++)

{

cout << " <-";

}

cout << endl;

}

int main()

{

Node \*head = NULL;

int value, choice = 0;

cout << "1) Show " << endl;

cout << "2) insert at start " << endl;

cout << "3) insert at end " << endl;

cout << "4) insert Inorder " << endl;

cout << "5) delete by value " << endl;

cout << "6) Exit " << endl;

do

{

cout << "enter choice :";

cin >> choice;

switch (choice)

{

case 1:

display(head);

break;

case 2:

cout << "Enter Value : ";

cin >> value;

InsertAtStart(head, value);

break;

case 3:

cout << "Enter Value : ";

cin >> value;

InsertAtlast(head, value);

break;

case 4:

cout << "Enter Value : ";

cin >> value;

insertInOrder(head, value);

break;

case 5:

cout << "Enter Value : ";

cin >> value;

deleteNodeByValue(head, value);

break;

default:

break;

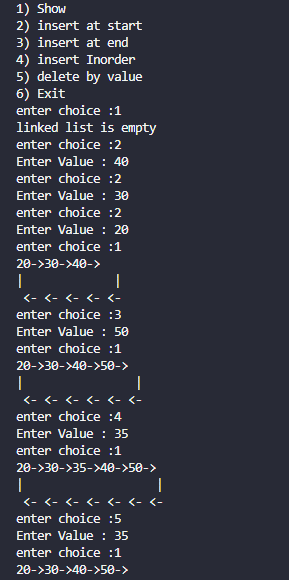
}

} while (choice != 6);

return 0;

}

**OUTPUT**



Time analysis

