8b)Design and implement a given type of (circular queue) queue in c (linked list

implementation) and demonstrate its working with suitable inputs. Display appropriate

messages in case of exceptions.

AIM: Implementation of circular queue using linked list.

ALGORITHM:

1. enQueue(value)

This function is used to insert an element into the circular queue. In a circular queue, the

new element is always inserted at the Rear position.

a. Create a new node dynamically and insert value into it.

b. Check if front==NULL, if it is true then front = rear = (newly created node)

c. If it is false then rear=(newly created node) and rear node always contains the

address of the front node.

2. deQueue()

This function is used to delete an element from the circular queue. In a queue, the element

is always deleted from the front position.

d. Check whether the queue is empty or not means front == NULL.

e. If it is empty then display Queue is empty. If queue is not empty then step 3

f. Check if (front==rear) if it is true then set front = rear = NULL else move the

front forward in queue, update address of front in rear node and return the

element.

Program:

#include <stdio.h> //header files

#include <stdlib.h>

struct node // creating a node

{

int data;

struct node \*next;

};

struct node \*front = -1;

struct node \*rear = -1;

void enqueue(int x) // function to enter the value to the circular queue

{

struct node \*newnode;

newnode = (struct node \*)malloc(sizeof(struct node));

newnode->data = x;

newnode->next = 0;

if (rear == -1)

{

front = rear = newnode;

rear->next = front;

}

else

{

rear->next = newnode;

rear = newnode;

rear->next = front;

}

}

void dequeue() // function to delete the value from the circular queue

{

struct node \*temp;

temp = front;

if ((front == -1) && (rear == -1)) // checking if the circular queue is empty

{

printf("Queue is empty");

}

else if (front == rear)

{

front = rear = -1;

free(temp);

}

else

{

front = front->next;

rear->next = front;

free(temp);

}

}

void display() // function to display the elements in circular queue

{

struct node \*temp;

temp = front;

if ((front == NULL) && (rear == NULL)) // checking if circular queue is empty

printf("\nQueue is Empty");

else

{

do

{

printf("\n%d", temp->data);

temp = temp->next;

} while (temp != front);

}

}

int main() // driver function

{

int choice, n, i, data;

while (1)

{

printf("\nSelect the operation\n");

printf("1.Insert\n2.Delete\n3.Display\n4.Exit\n");

scanf("%d", &choice);

switch (choice)

{

case 1:

printf("\nEnter the number of data:");

scanf("%d", &n);

printf("\nEnter your data");

i = 0;

while (i < n)

{

scanf("%d", &data);

enqueue(data);

i++;

}

break;

case 2:

dequeue();

break;

case 3:

display();

break;

case 4:

exit(0);

default:

printf("Invalid choice");

break;

}

}

return 0;

}

Output:

