Ex. No. 3c **Queue Using Linked List**

Date:

**Aim:**

To implement queue operations using linked list

**Algorithm:**

1. Start

2. Define a singly linked list node for stack

3. Create Head node

4. Display a menu listing stack operations

5. Accept choice

6. If choice = 1 then

Create a new node with data

Make new node point to first node

Make head node point to new node

Else If choice = 2 then

Make temp node point to first node

Make head node point to next of temp node

Release memory

Else If choice = 3 then

Display stack elements starting from head node till null

7. Stop

**Program:**

/\* Queue using Single Linked List \*/

#include <stdio.h>

#include <stdlib.h>

struct node

{

int label;

struct node \*next;

};

void main()

{

int ch=0;

int k;

struct node \*h, \*temp, \*head;

/\* Head node construction \*/

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

head->next = NULL;

while(1)

{

printf("\n Queue using Linked List \n");

printf("1->Insert ");

printf("2->Delete ");

printf("3->View ");

printf("4->Exit \n");

printf("Enter your choice : ");

scanf("%d", &ch);

switch(ch)

{

case 1:

/\* Create a new node \*/

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

printf("Enter label for new node : ");

scanf("%d", &temp->label);

/\* Reorganize the links \*/

h = head;

while (h->next != NULL)

h = h->next;

h->next = temp;

temp->next = NULL;

break;

case 2:

/\* Delink the first node \*/

h = head->next;

head->next = h->next;

printf("Node deleted \n");

free(h);

break;

case 3:

printf("\n\nHEAD -> ");

h=head;

while (h->next!=NULL)

{

h = h->next;

printf("%d -> ",h->label);

}

printf("NULL \n");

break;

case 4:

exit(0);

}

}

}

**Result:**

Thus insert and delete operations of a queue was demonstrated using linked list.