Navlakhi®

Queues: FIFO

Methodology and Program

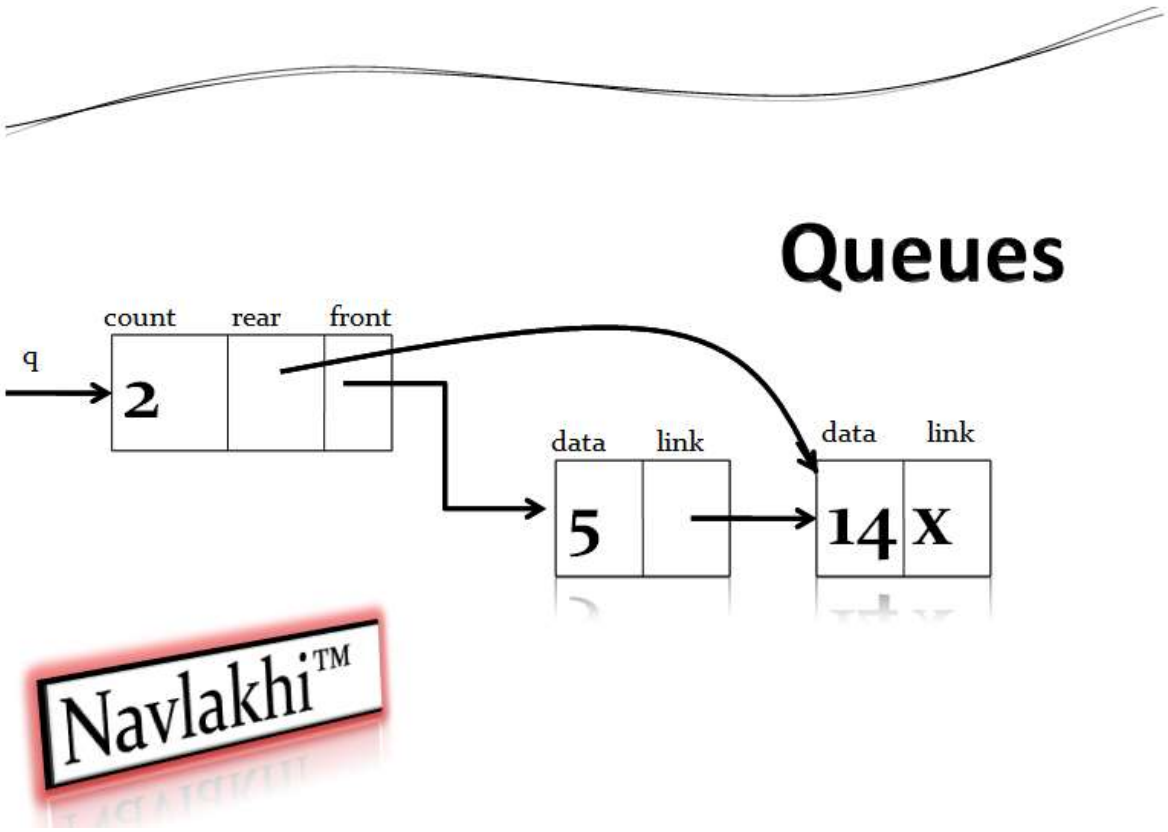
By Abhishek Navlakhi

Semester 3: Data Structures

In Queue the insertion (enqueue) is at the rear of the queue & the removal (dequeue) is from the front. Hence we maintain 2 pointers, the front pointer and the rear pointer to facilitate the addition & removal process. Note that 'enqueue' will always add a new node at the rear, hence rear should point to the new node. Similarly, dequeue will remove the node from front & cause the second node to become the front. Special case is which we have 0 nodes & we are adding the first node the front should also be made to point to it & similarly when we have only one node & we want to dequeue it then the rear will also have to be forced to 'null'.

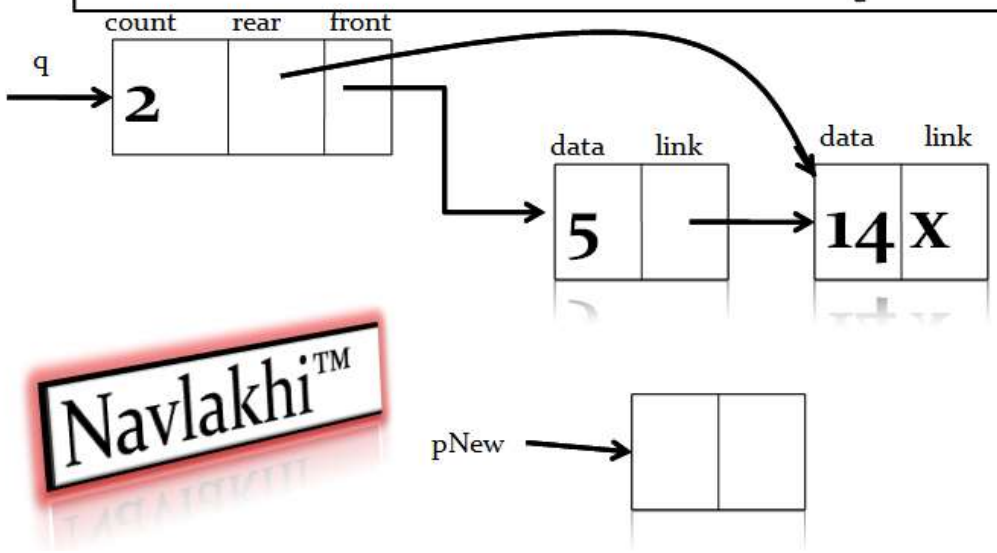
Let's lay down a few programming basics

ADDITION OF DATA (ENQUEUE)



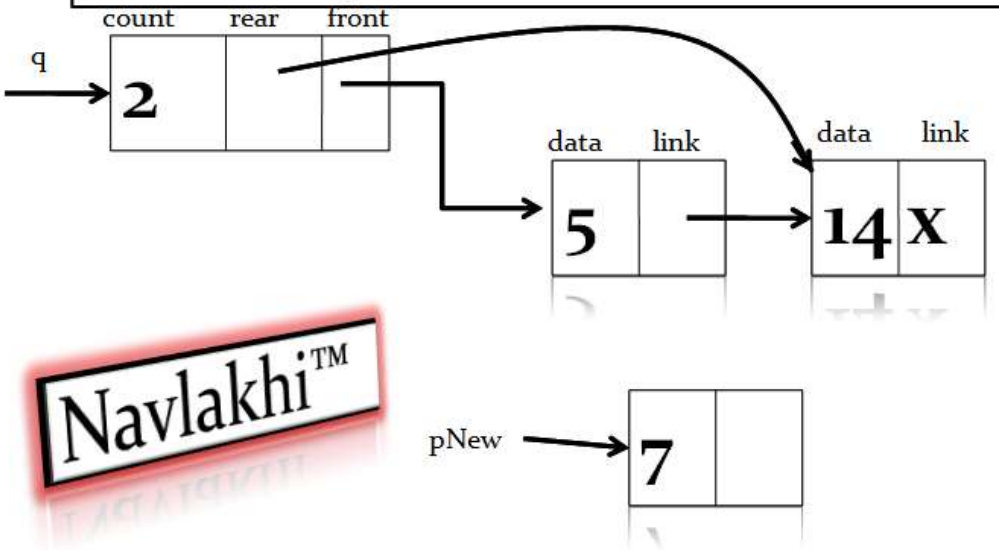
Queues- Adding 7 ENQUEUE

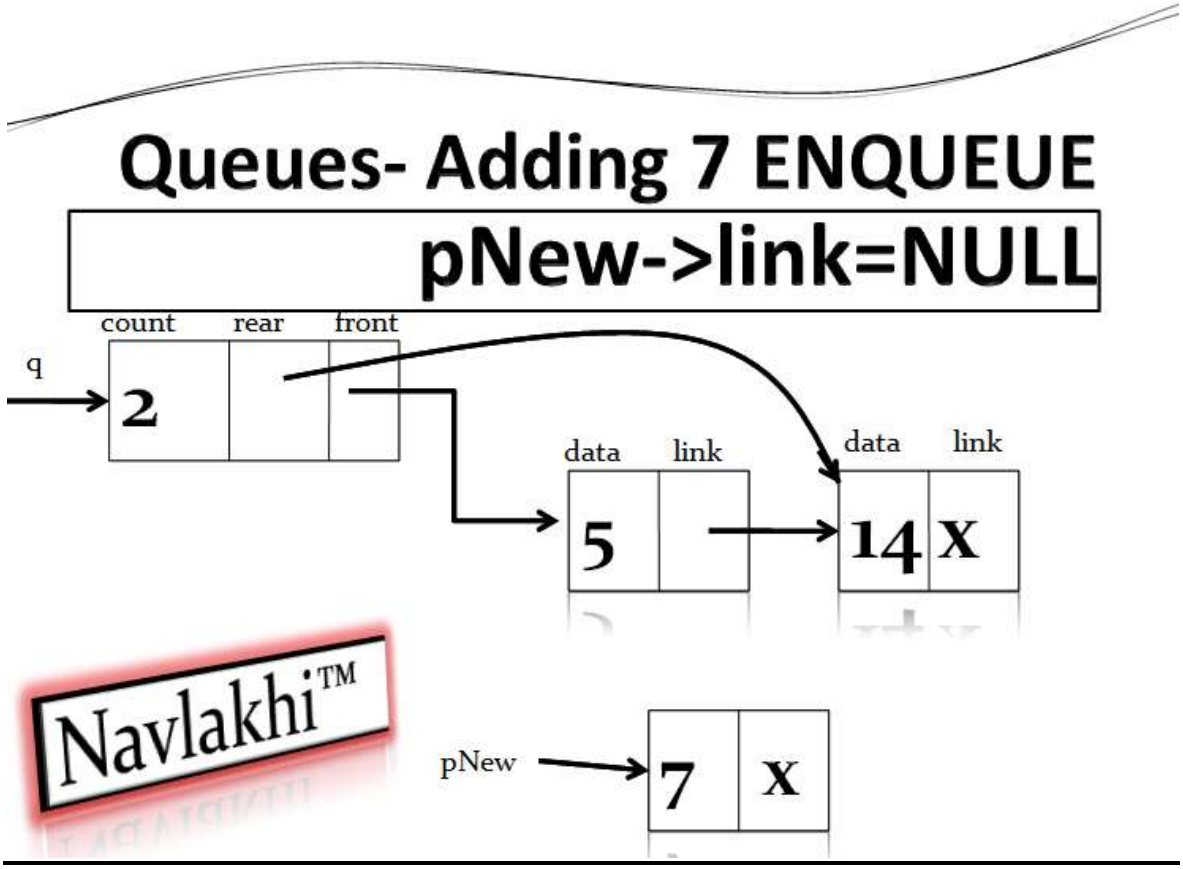
malloc pNew

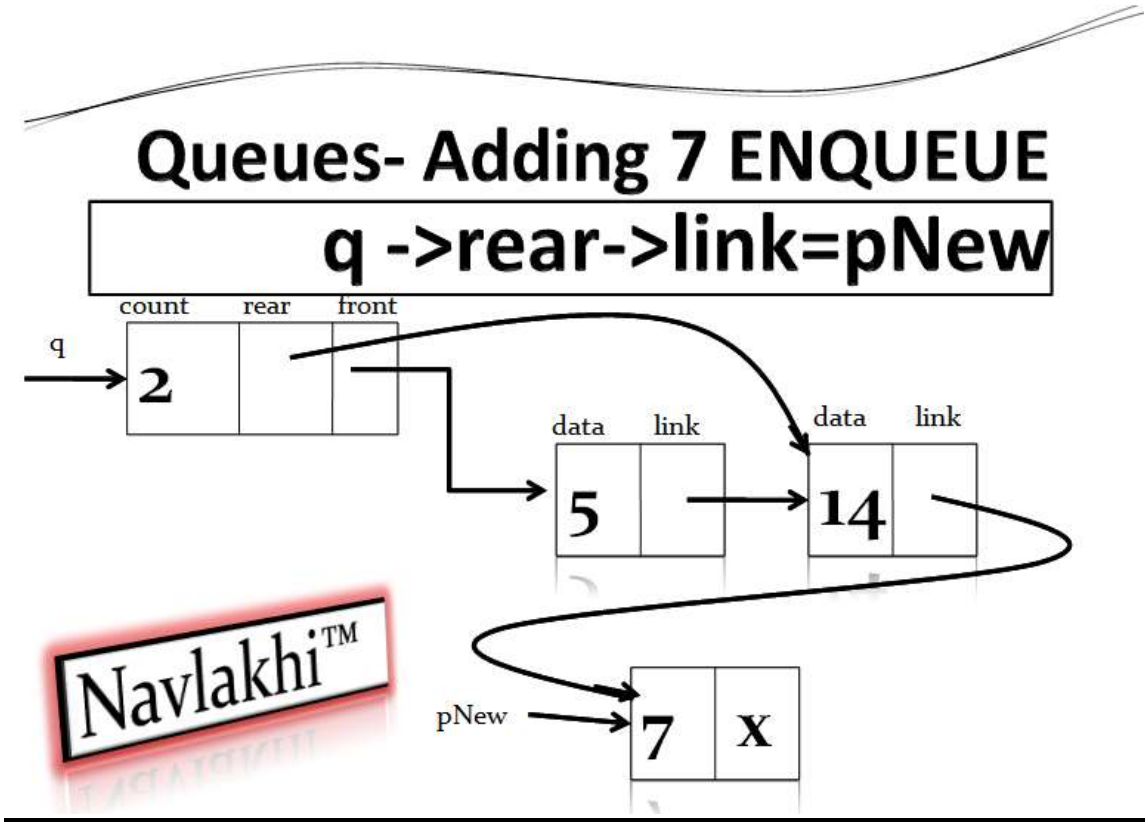


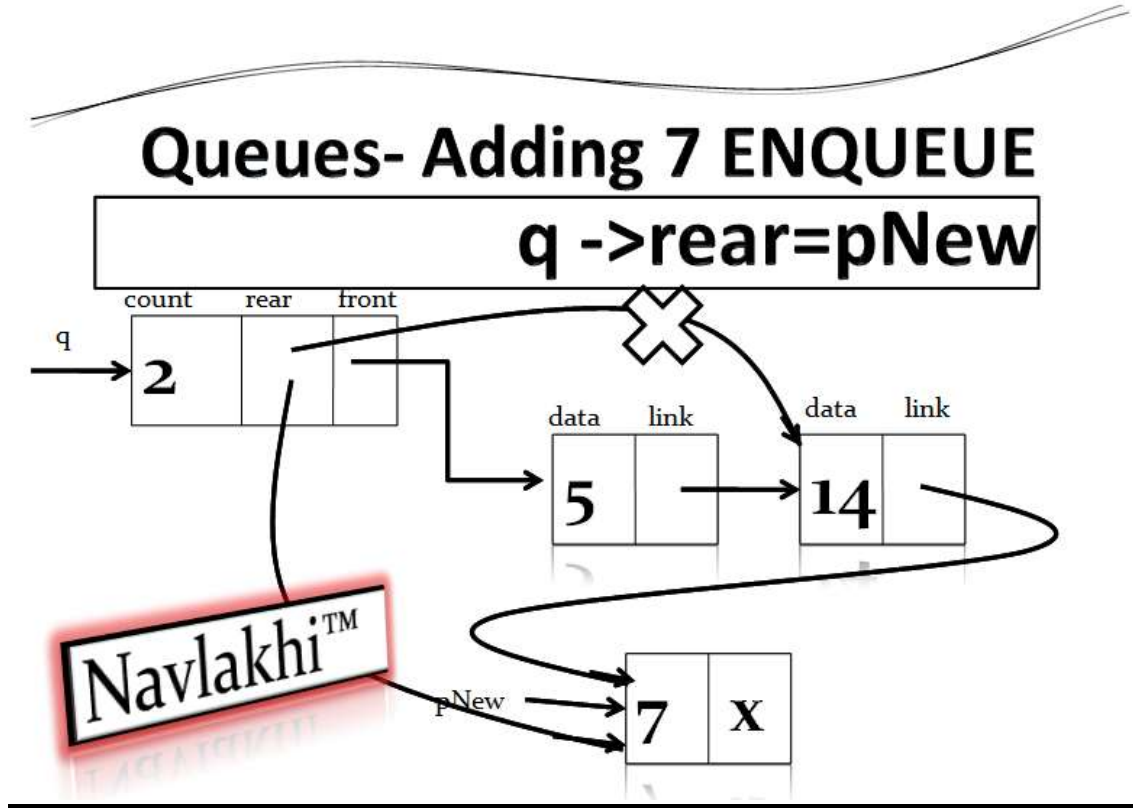
Queues- Adding 7 ENQUEUE

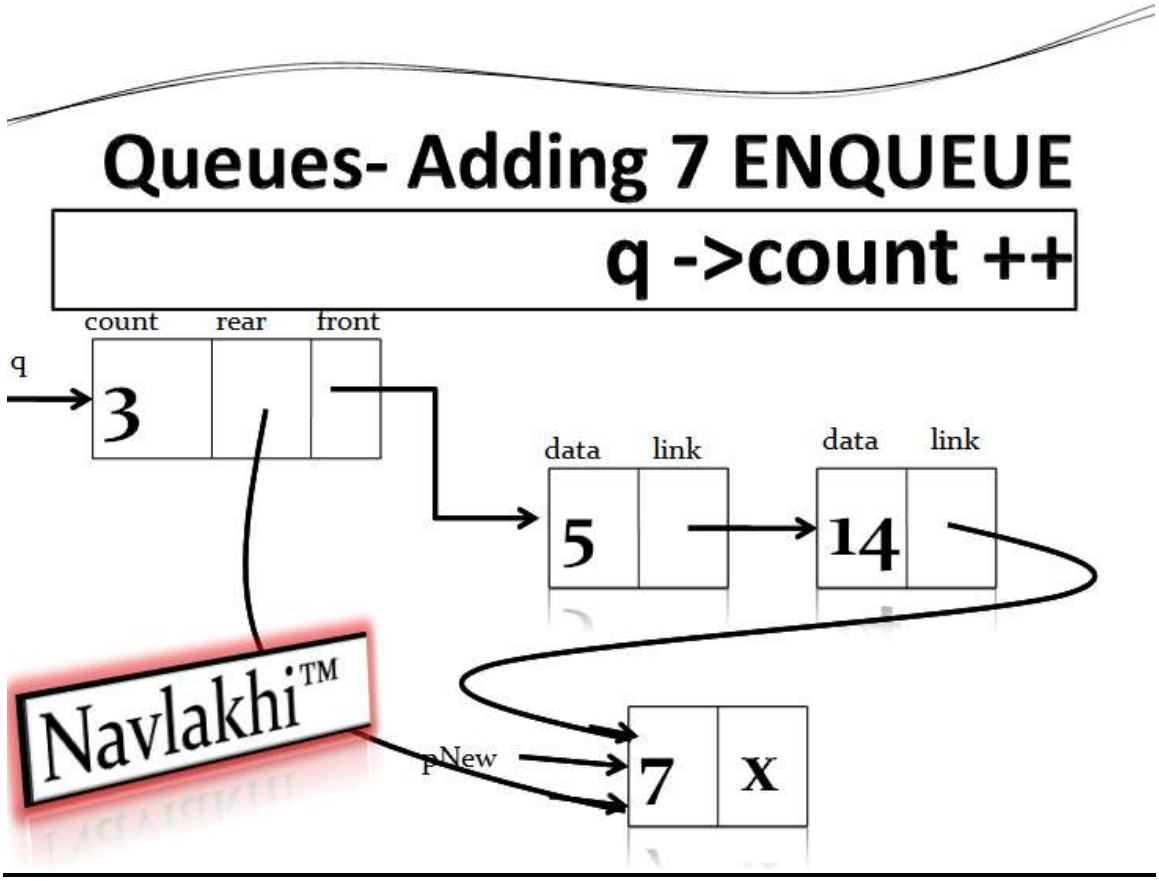
pNew->data=dataIn




Queues- Adding 7 ENQUEUE
pNew->link=NULL
count
rear
front
q
2
data
link
5
data
link
14
X
Navlakhi™
pNew
7
X

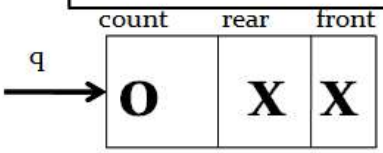

Queues- Adding 7 ENQUEUE
q ->rear->link=pNew
count
rear
front
q
2
data
link
5
data
link
14
Navlakhi™
pNew
7
X

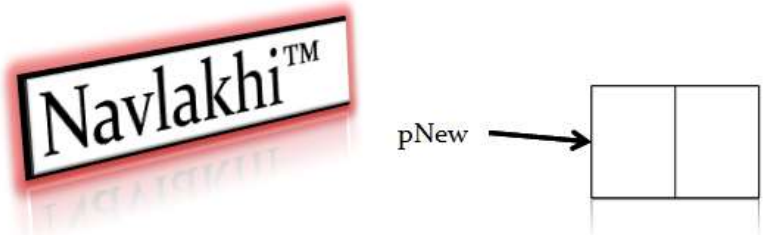

Queues- Adding 7 ENQUEUE
q ->rear=pNew
count
rear
front
q
2
data
link
5
data
link
14
pNew
7
X
Navlakhi™


Queues- Adding 7 ENQUEUE
q ->count ++
count
rear
front
q
3
data
link
5
data
link
14
pNew
7
X
Navlakhi™

Queues-SPECIAL CASE

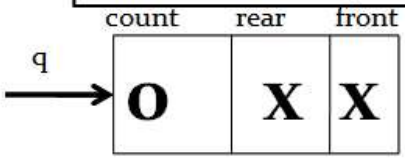
malloc pNew





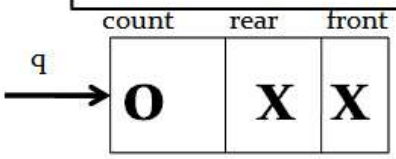
Queues-SPECIAL CASE

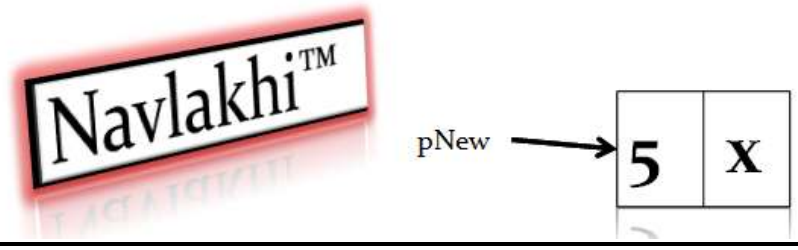
pNew->data=dataIn



Queues-SPECIAL CASE

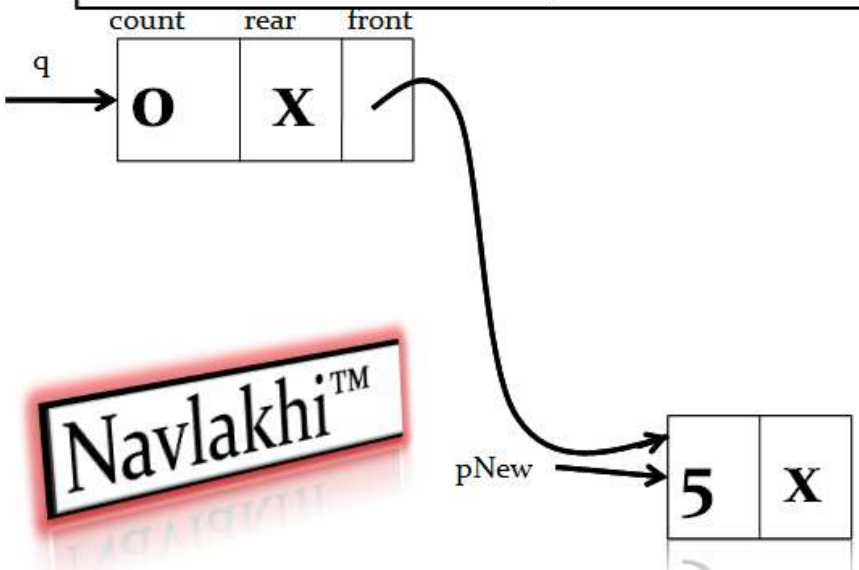
pNew->link=NULL

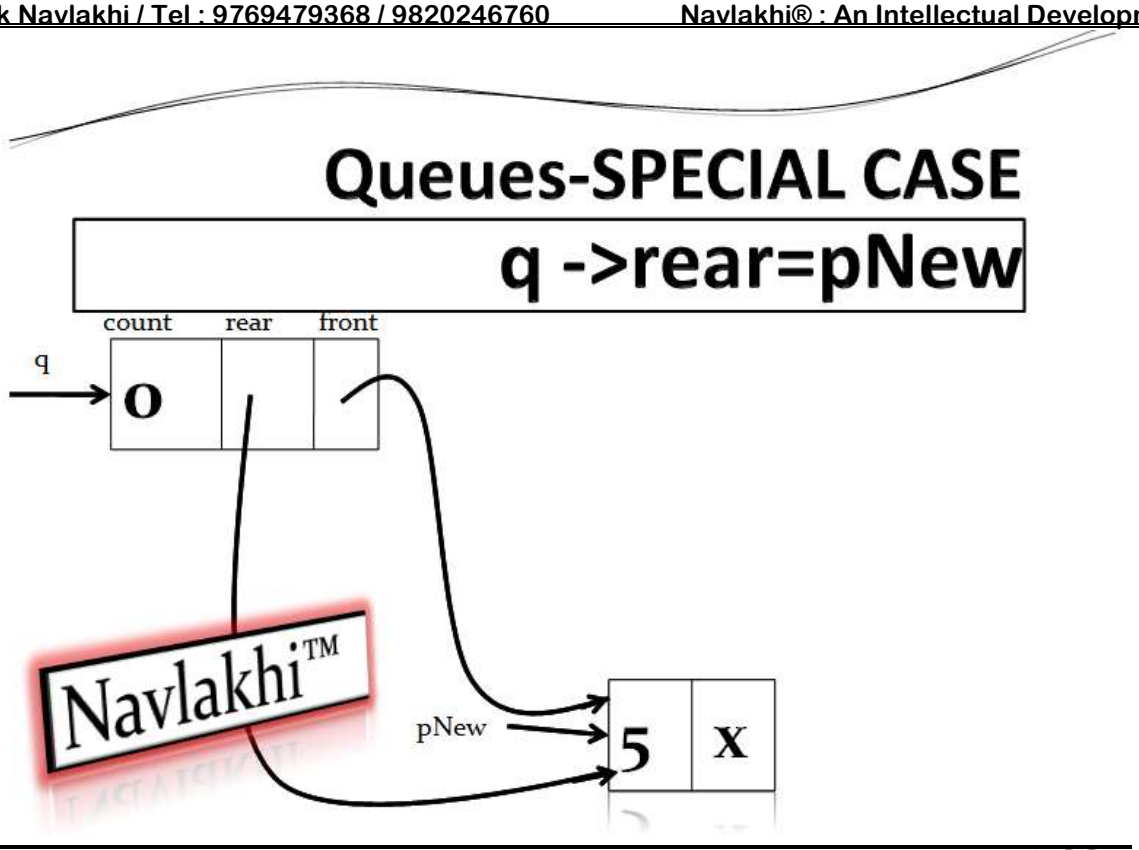


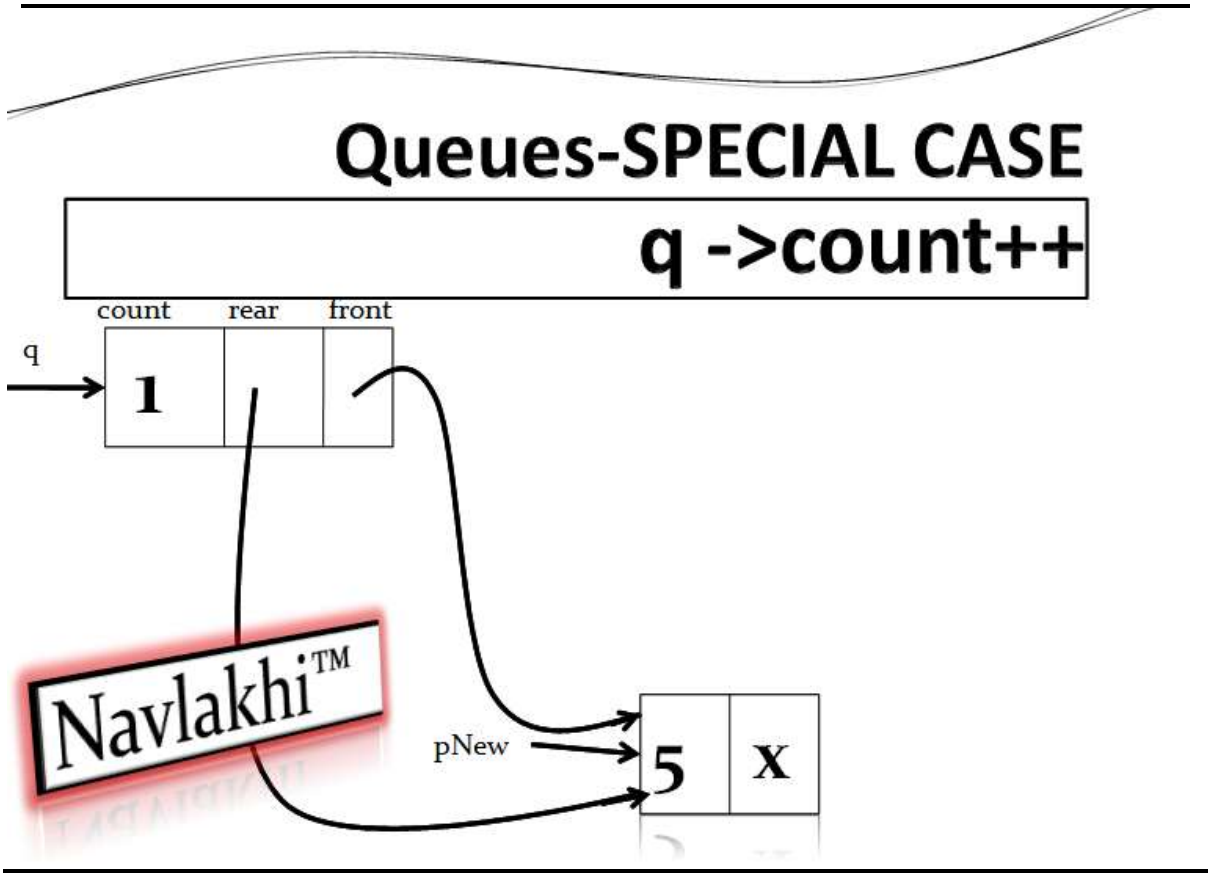


Queues-SPECIAL CASE

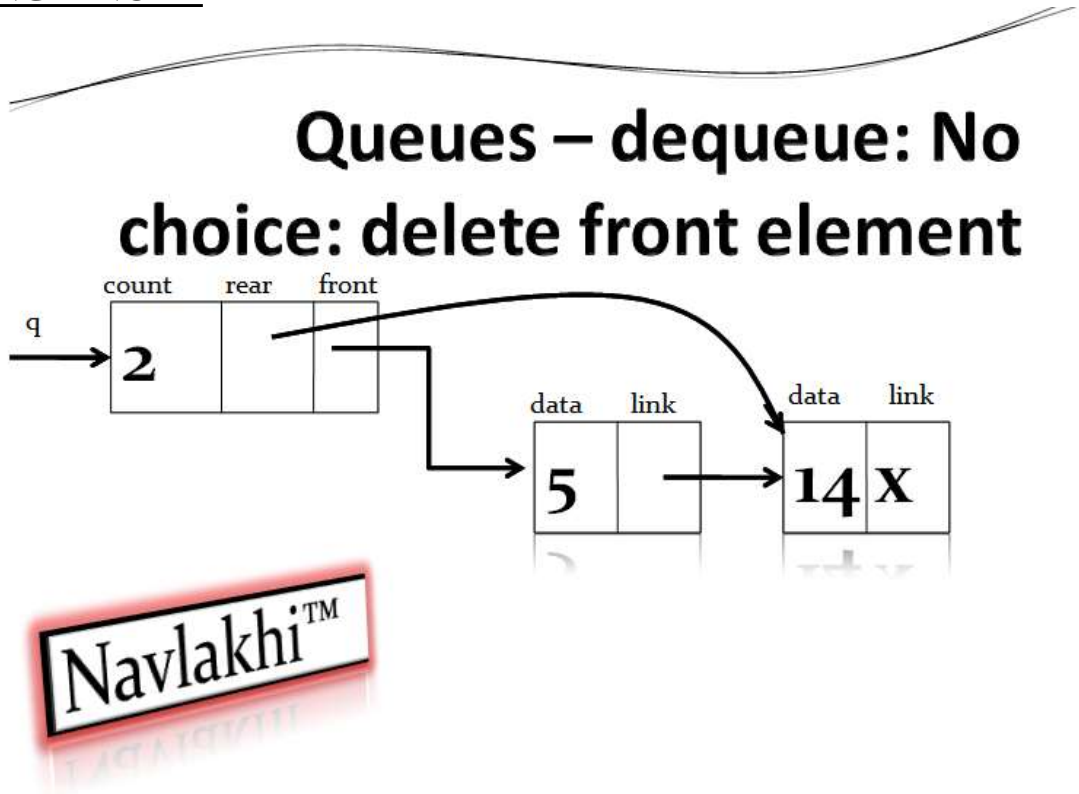
q ->front=pNew

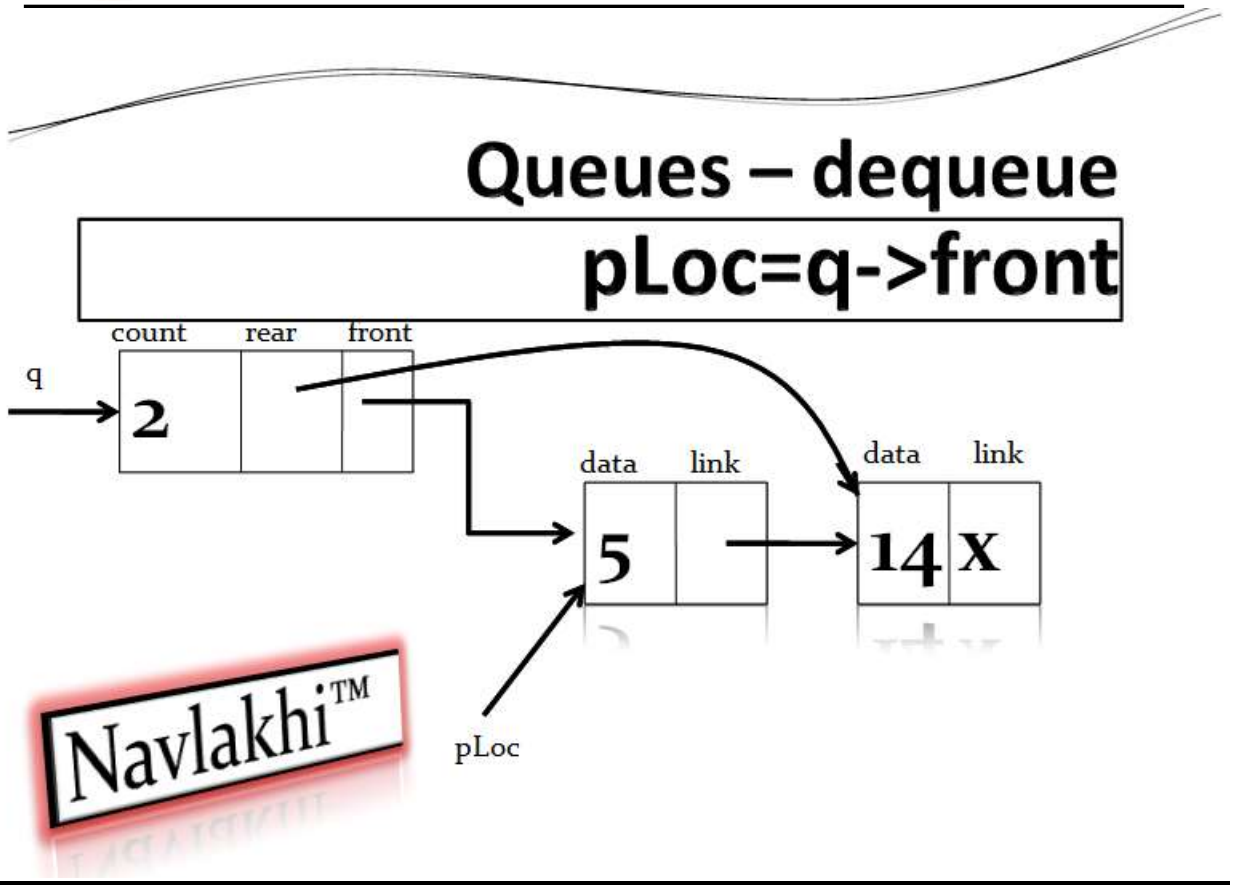


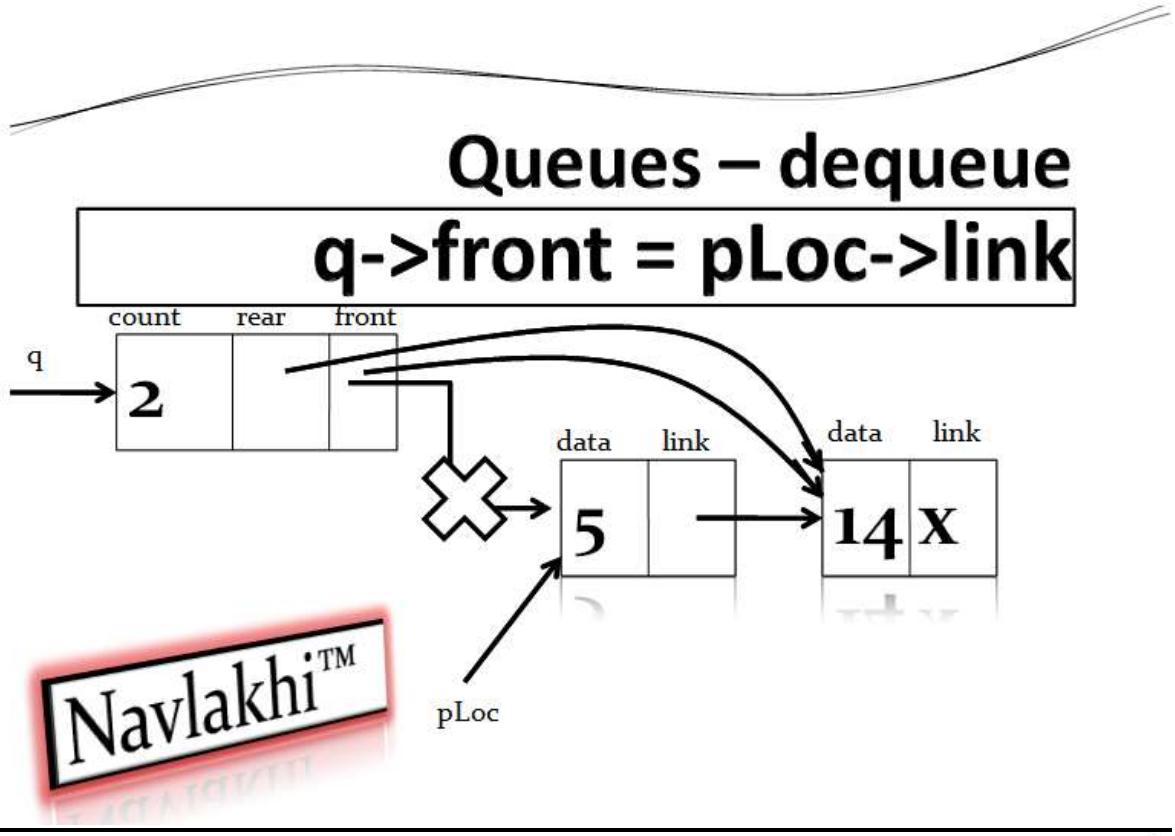

Queues-SPECIAL CASE
q ->rear=pNew
count
rear
front
q
0
Navlakhi™
pNew
5
X

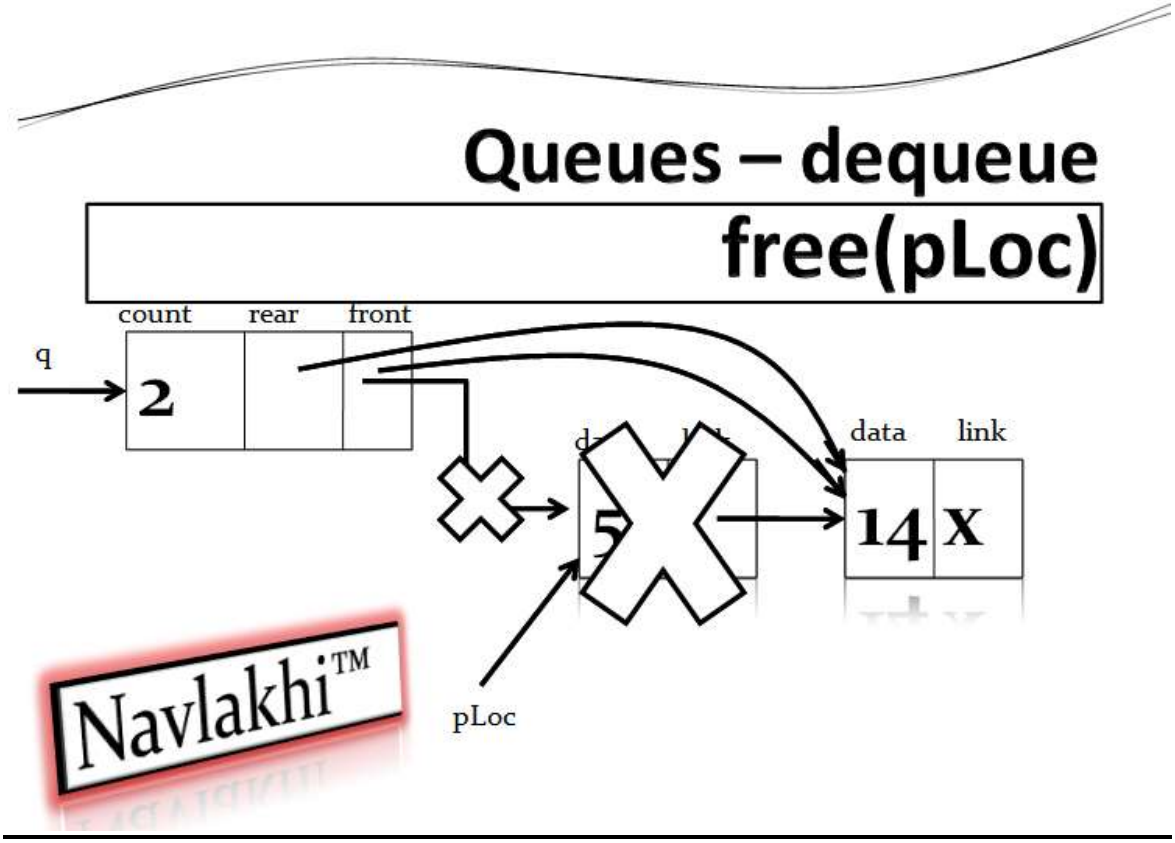

Queues-SPECIAL CASE
q ->count++
count
rear
front
q
1
Navlakhi™
pNew
5
X

DELETING A NODE



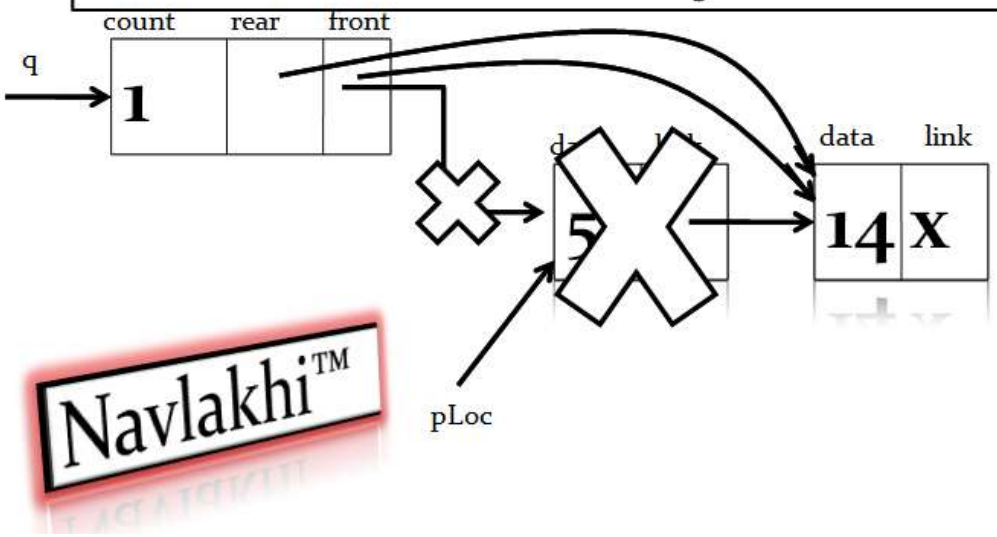



Queues – dequeue
q->front = pLoc->link
count
rear
front
q
2
data
link
data
link
5
14
X
pLoc
Navlakhi™


Queues – dequeue
free(pLoc)
count
rear
front
q
2
data
link
5
14
X
pLoc
Navlakhi™

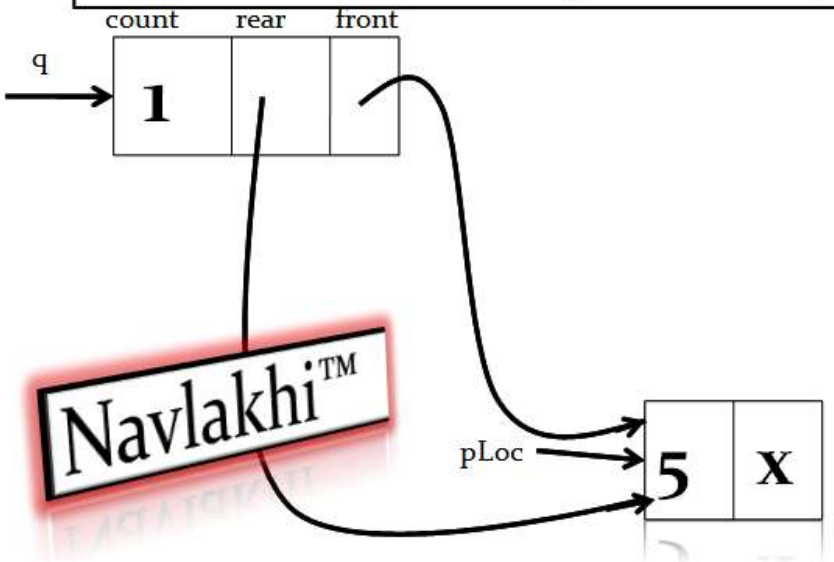
Queues – dequeue

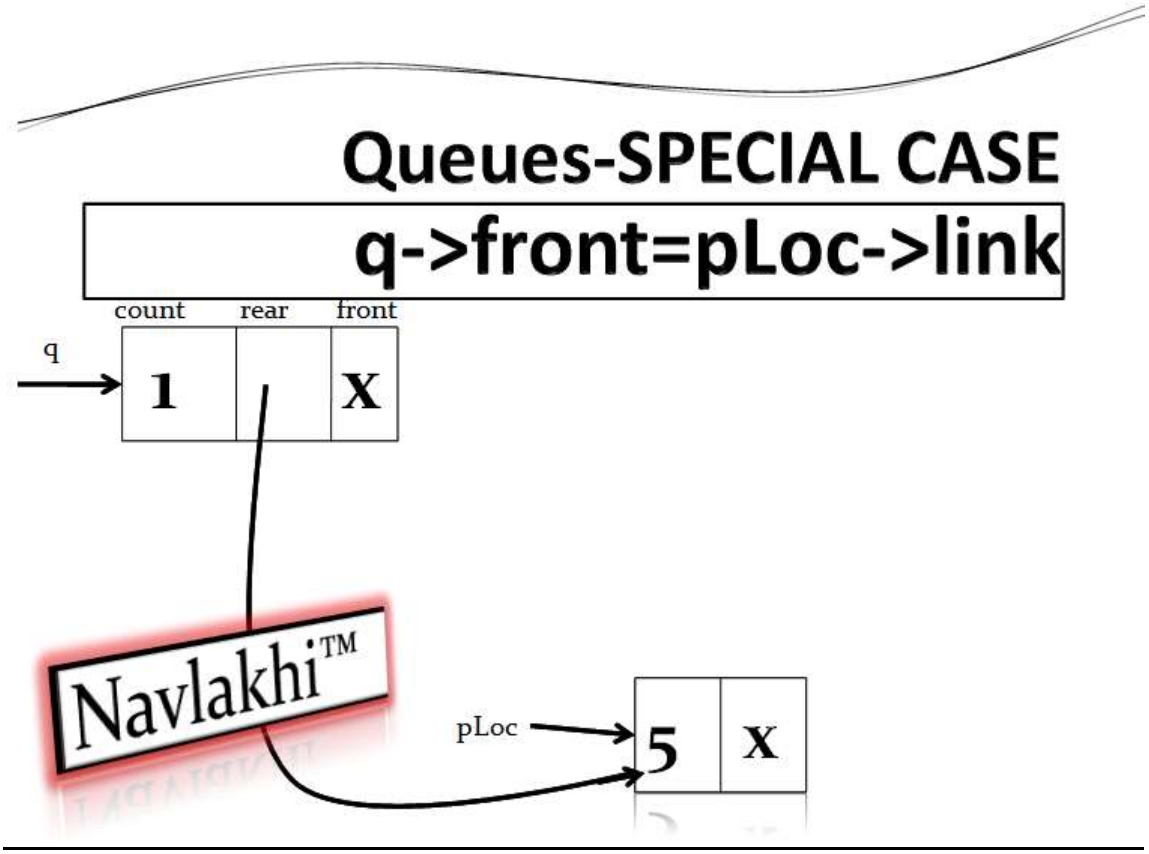
q->count --

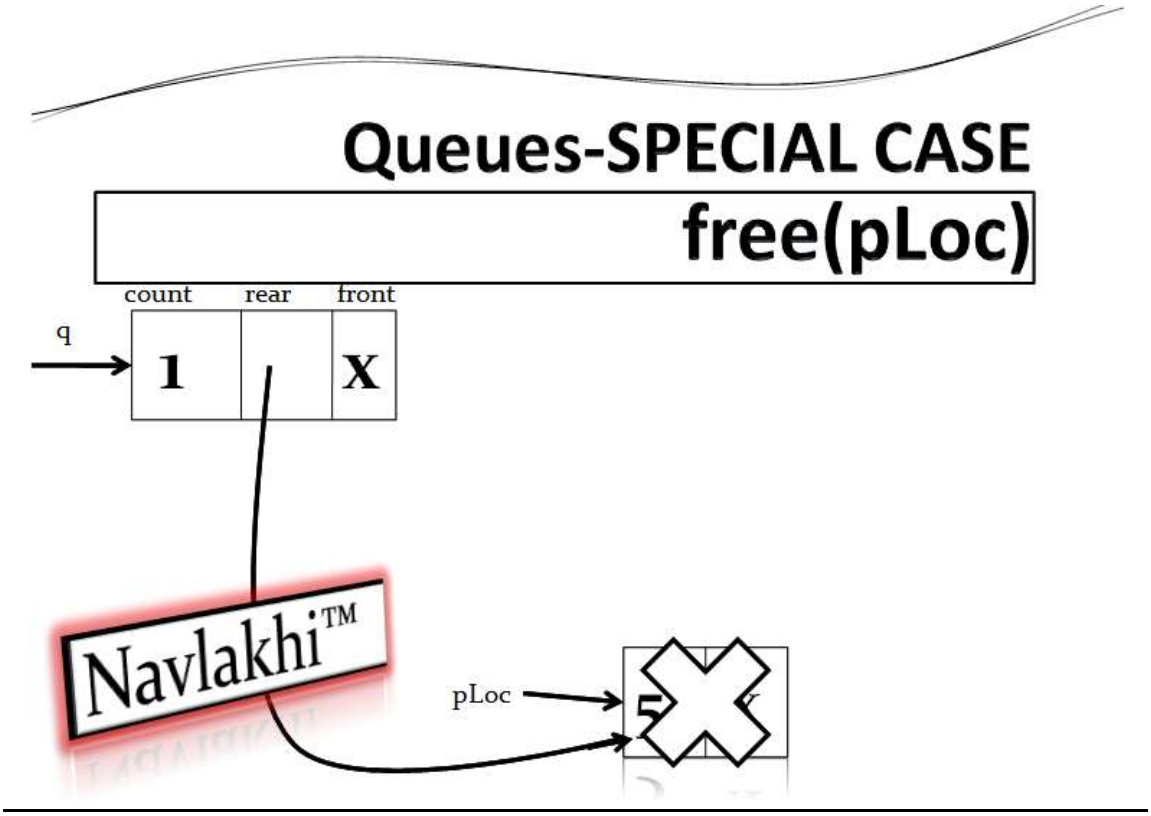


Queues-SPECIAL CASE

pLoc=q->front

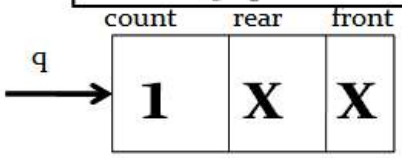


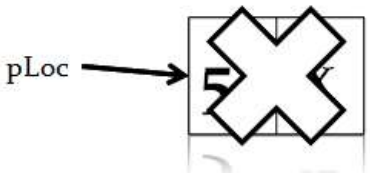

Queues-SPECIAL CASE
q->front=pLoc->link
count
rear
front
q
1
X
Navlakhi™
pLoc
5
X


Queues-SPECIAL CASE
free(pLoc)
count
rear
front
q
1
X
Navlakhi™
pLoc

Queues-SPECIAL CASE

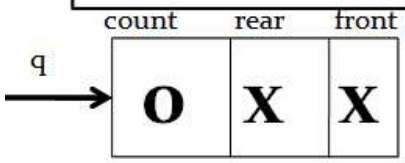
```
if (q->count==1) q->rear=NULL
```

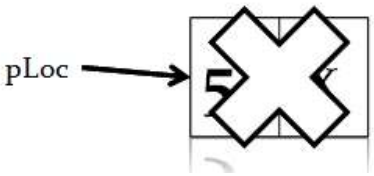




Queues-SPECIAL CASE

```
q->count--
```





Program

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


struct node
{
int data;
struct node *link;
};


struct queue
{
int count;
struct node *front;
struct node *rear;
}*q;
```

```
int queueFront(int *dataPtr)
{
if (q->count==0) return 0;

*dataPtr=q->front->data;
return 1;
}


int queueRear(int *dataPtr)
{
if (q->count==0) return 0;

*dataPtr=q->rear->data;
return 1;
}


void destroyQueue( )
{
struct node *pLoc;
while (q->count!=0)
{
        pLoc=q->front;
        q->front=q->front->link; /* OR q->front=pLoc->link */
        q->count-=1;
        free(pLoc);
}
free(q);
}

int fullQueue( )
{
struct node *temp;
temp=(struct node*)malloc(sizeof(struct node*));
if (temp==NULL)    return 1;
else { free(temp);  return 0; }
}
```

```
int enqueue(int dataIn)
{
	struct node *pNew;

	pNew=(struct node *)malloc(sizeof(struct node));

	if (pNew!=NULL)
	{
		pNew->data=dataIn;
		pNew->link=NULL;
		if (q->count==0)
			q->front=pNew;
		else
			q->rear->link=pNew;

		q->rear=pNew;
		q->count+=1;
		return 1;
	}
	else
		return 0;
}

int dequeue(int *dataPtr)
{
	struct node *pLoc;
	if (q->count==0) return 0;

	pLoc=q->front;
	*dataPtr=pLoc->data; /****OR *dataPtr=q->front->data   ***/

	if (q->count==1) q->rear=NULL;

	q->front=q->front->link; /*OR q->front=pLoc->link */
	q->count-=1;
	free(pLoc);
	return 1;
}
```

```
int menu( )
{
int choice;
printf("\n\n\t\t M E N U \n");
printf("\n\n\t\t ========\n");
printf("1. Add Data To Queue (Enqueue)\n");
printf("2. Remove Data From Queue (Dequeue)\n");
printf("3. View Front element\n");
printf("4. View Rear element\n");
printf("5. View Count\n");
printf("6. Check if Memory FULL\n");
printf("7. Exit\n");

printf("\n Feed in your choice: ");
scanf("%d",&choice);

return choice;
}



void  createQueue( )
{
        q =(struct queue *)malloc(sizeof(struct queue));
        if (q ==NULL)
        {
                printf("Insufficient memory.....\n");
                exit(1);
        }
        q ->front= NULL;
        q ->rear=NULL;
        q ->count=0;
}
```

```
void main( )
{
int choice,dataIn,dataOut,success;

createQueue( );
do
{
        choice=menu( );

        switch(choice)
        {
        case 1:         printf("feed in Data to insert: ");
                scanf("%d",&dataIn);
                success=enqueue(dataIn);
                if(success) printf("Data inserted successfully\n");
                else printf("data Insertion Failed.. Insufficient memory..\n");
                break;
        case 2:         success=dequeue(&dataOut);
                if (success) printf("Data successfully Deleted = %d\n",dataOut);
                else printf("Data Not present\n");
                break;
        case 3: success=queueFront(&dataOut);
                if (success) printf("Data at the front of queue= %d\n",dataOut);
                else printf("No data in queue\n");
                break;
        case 4: success=queueRear(&dataOut);
                if (success) printf("Data at the front of queue= %d\n",dataOut);
                else printf("No data in queue\n");
                break;
        case 5: printf("Number of Data Items in Queue = %d\n",q->count);
                break;
        case 6:         success=fullQueue();
                if (success) printf("Memory FULL !!!! ");
                else printf("Memory not FULL...\n");
                break;
        }
}while(choice!=7);
destroyQueue( );
}
```

```
case 3:
if(q->count==0)
printf("Nodata");
else
printf("%d",
 q->front->data);
break;
```

```
case 4:
if(q->count==0)
printf("Nodata");
else
printf("%d",
 q->rear->data);
break;
```

HOME OF EDUCATION

Navlakhi®

www.navlakhi.com
Home of Education

DATA Structures@ Navlakhi's