Navlakhi®

Doubly Linked List

Methodology and Program

By Abhishek Navlakhi

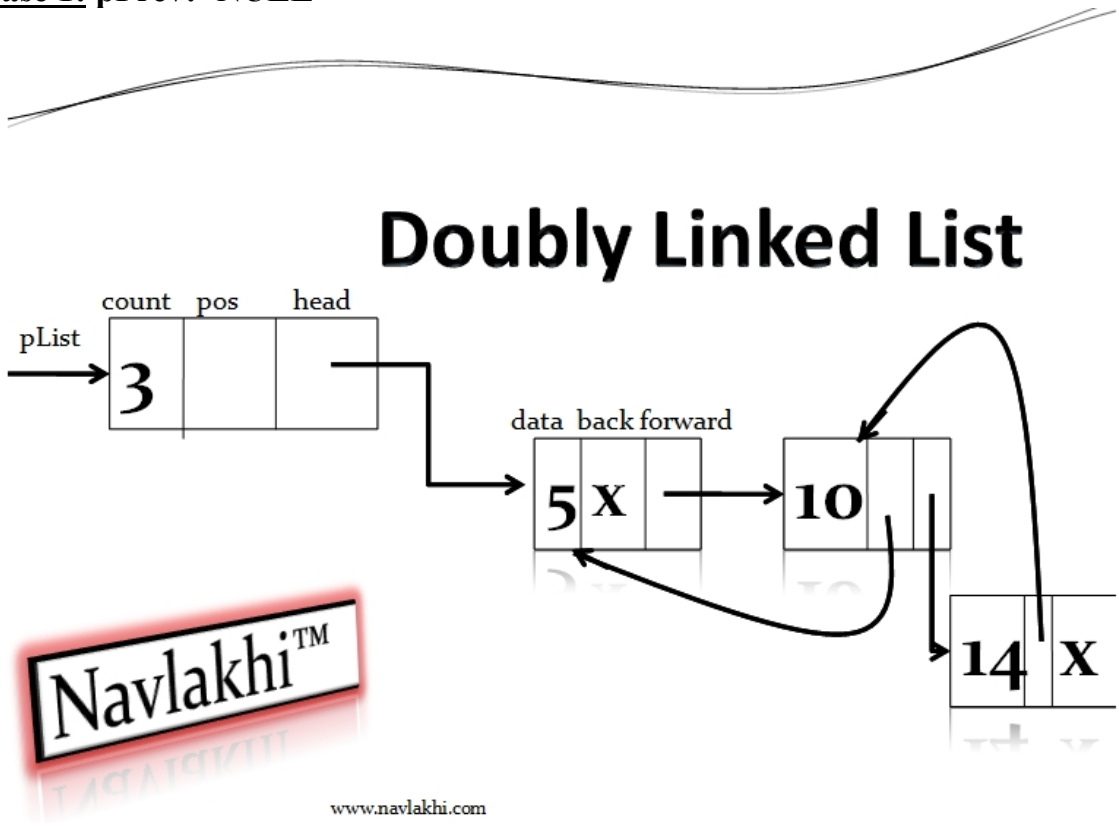
Semester 3: Data Structures

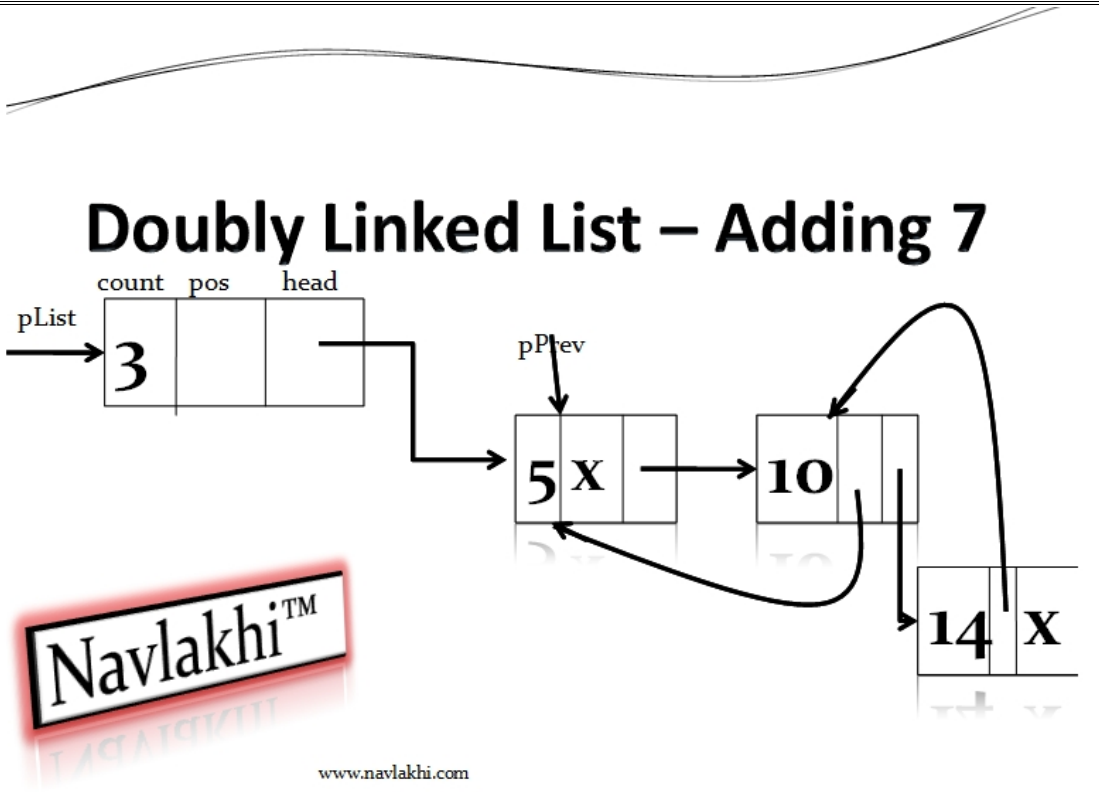
Doubly Linked list is similar to the Linked list except each Node has a forward & a back ward link. This helps in traversing in either direction.

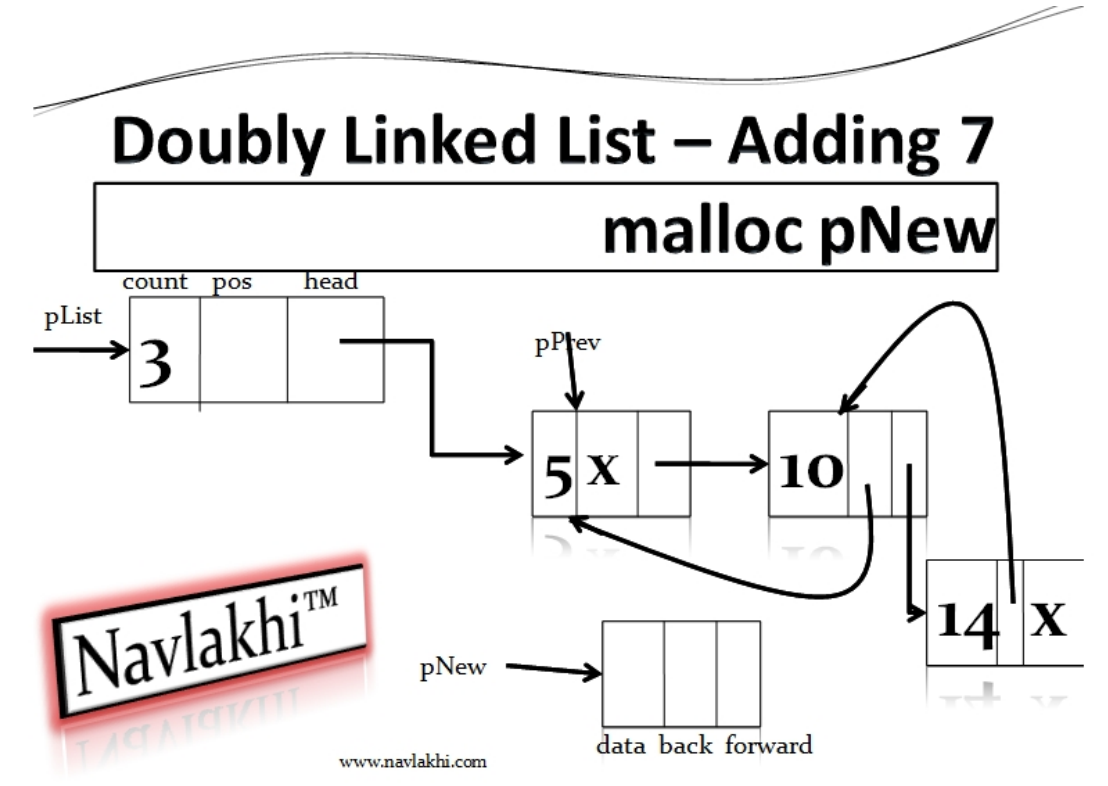
Let's lay down a few programming basics

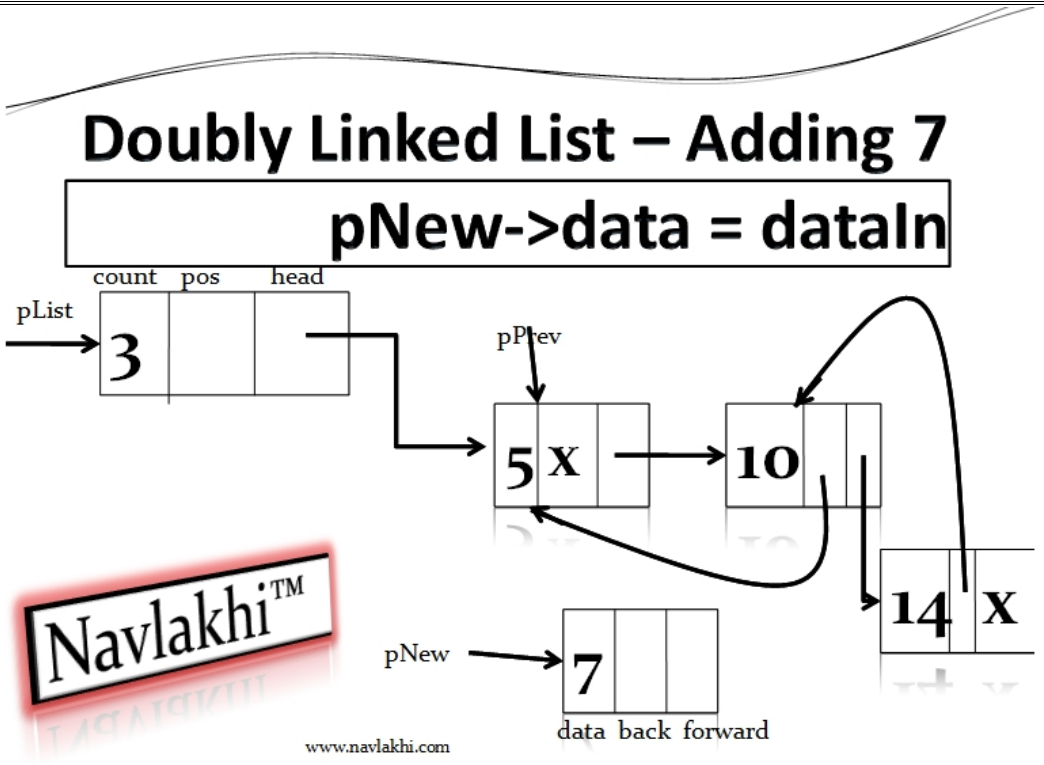
ADDITION OF DATA

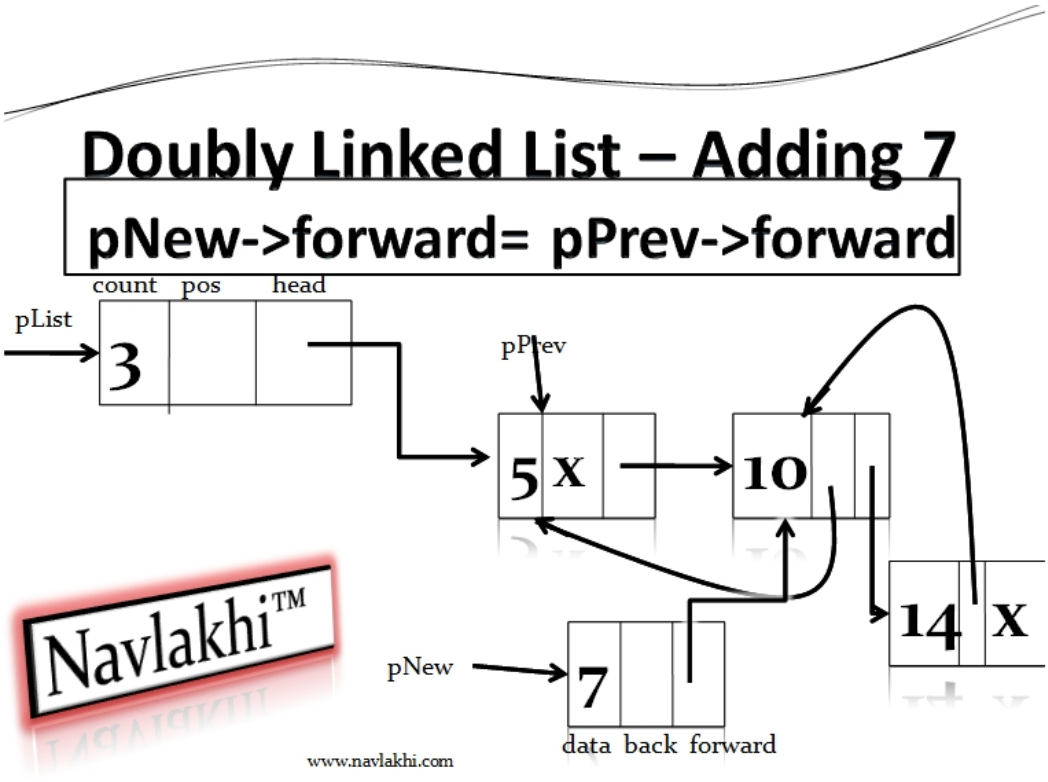
Case 1: pPrev!=NULL

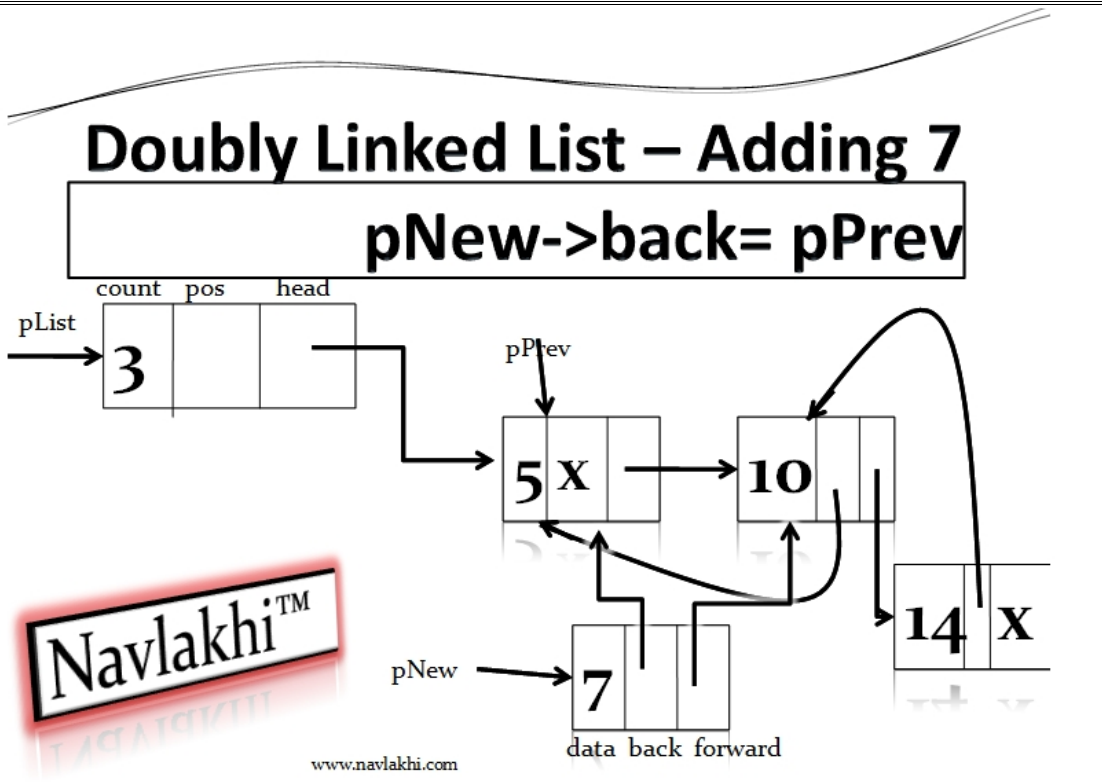


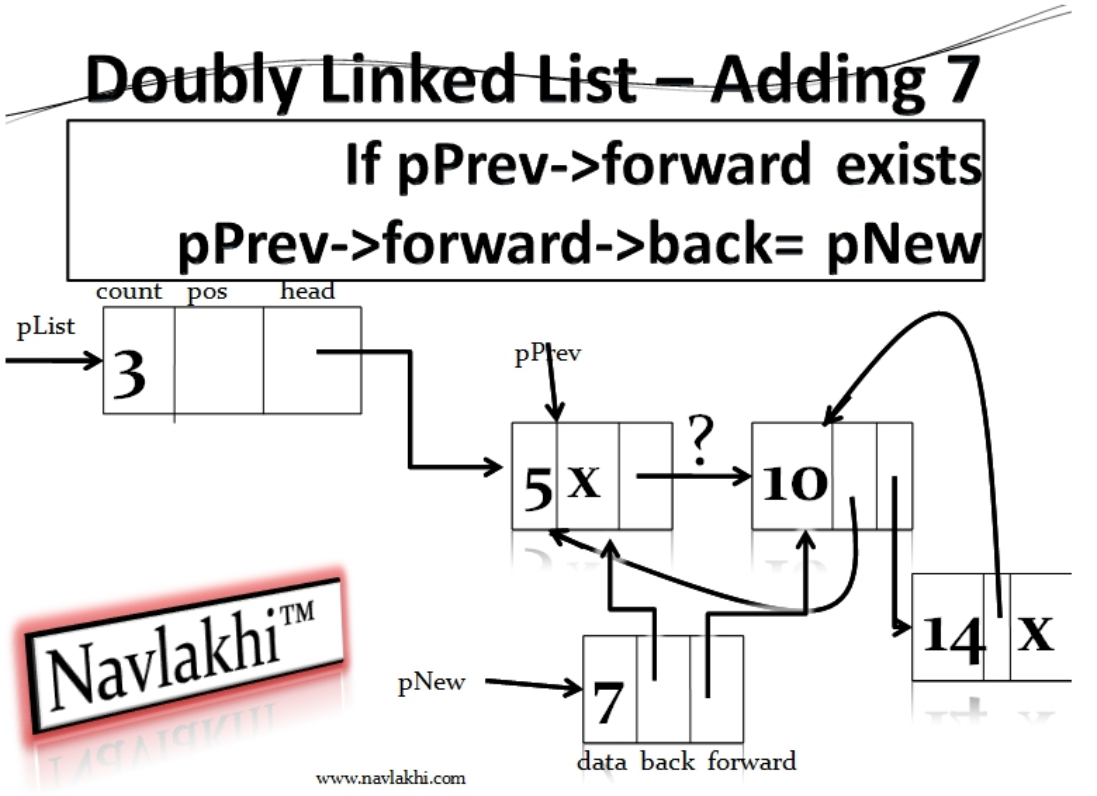

Doubly Linked List – Adding 7
count pos head
pList
3
pPrev
5 X
10
14 X
Navlakhi™
www.navlakhi.com

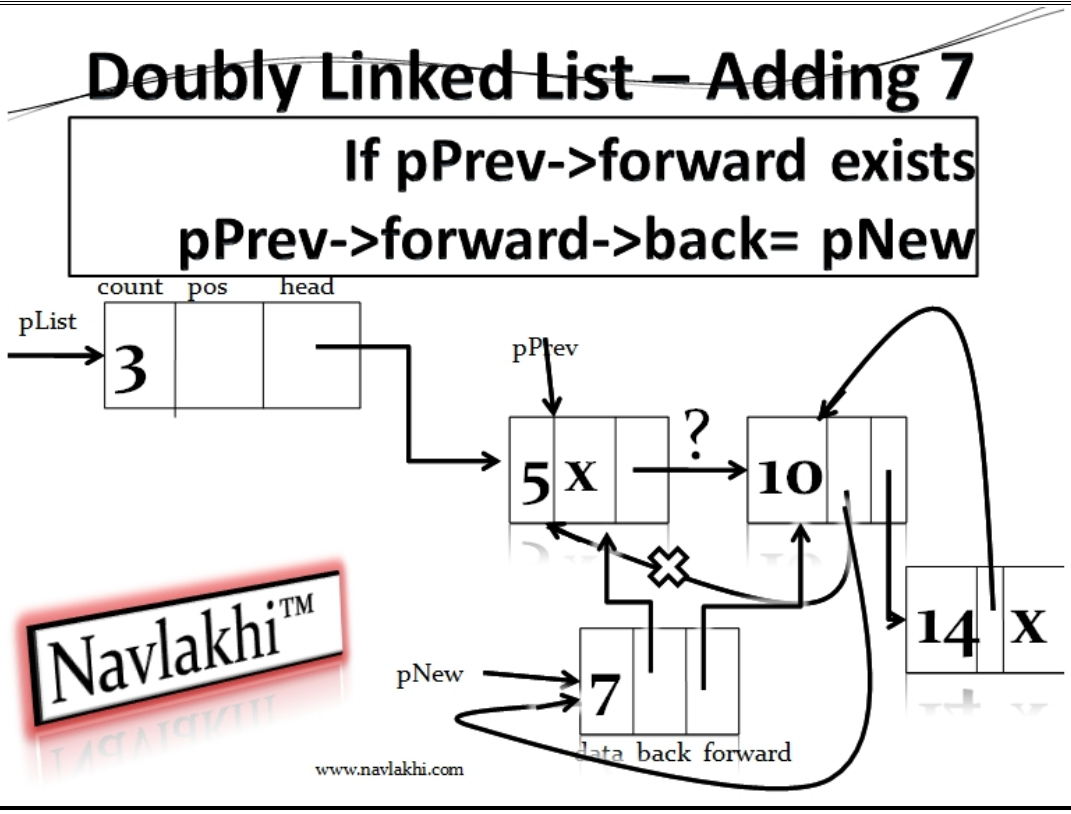

Doubly Linked List – Adding 7
malloc pNew
count pos head
pList
3
pPrev
5 X
10
14 X
Navlakhi™
pNew
data back forward
www.navlakhi.com

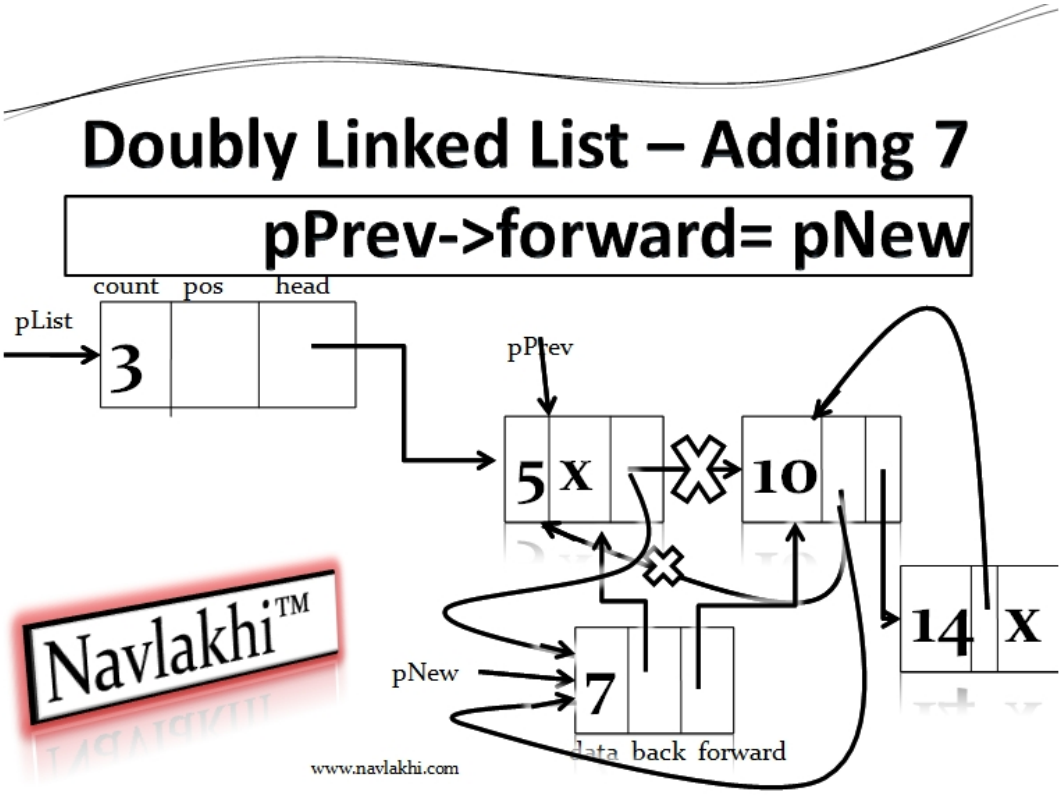

Doubly Linked List – Adding 7
pNew->data = dataIn
count pos head
pList
3
pPrev
5 X
10
14 X
pNew
7
data back forward
Navlakhi™
www.navlakhi.com

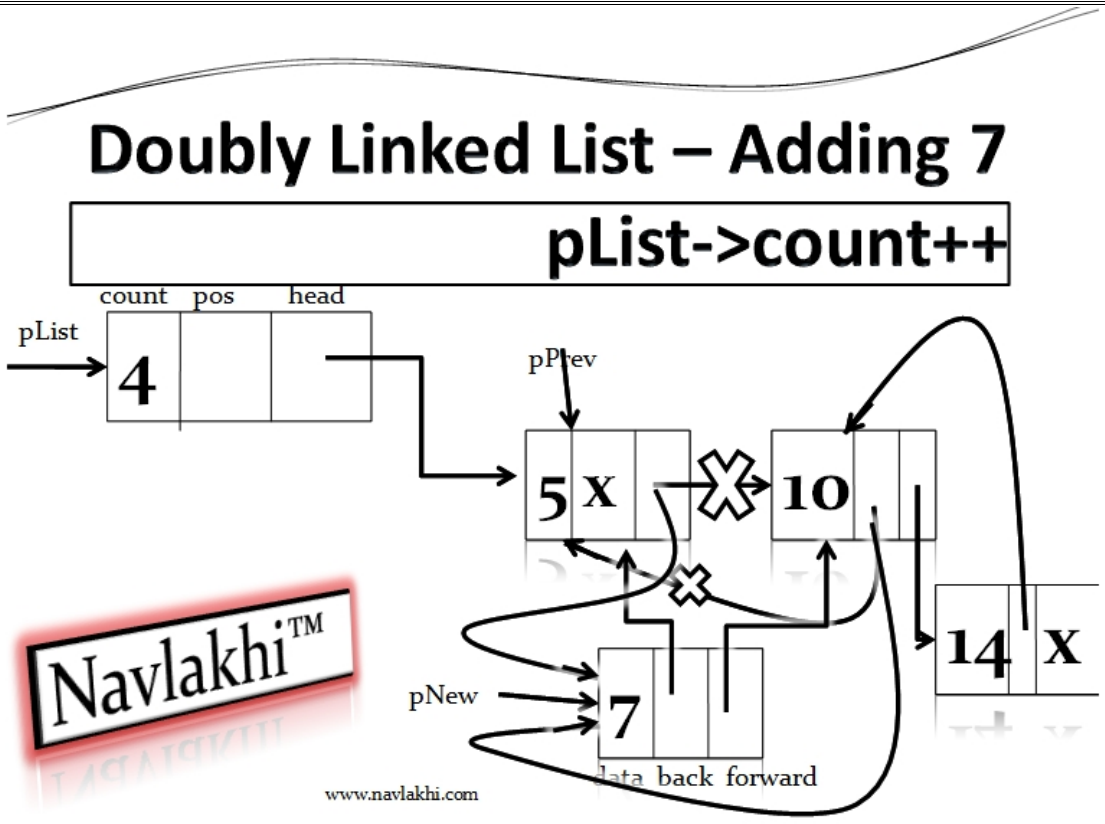

Doubly Linked List – Adding 7
pNew->forward= pPrev->forward
count pos head
pList
3
pPrev
5 X
10
14 X
pNew
7
data back forward
Navlakhi™
www.navlakhi.com

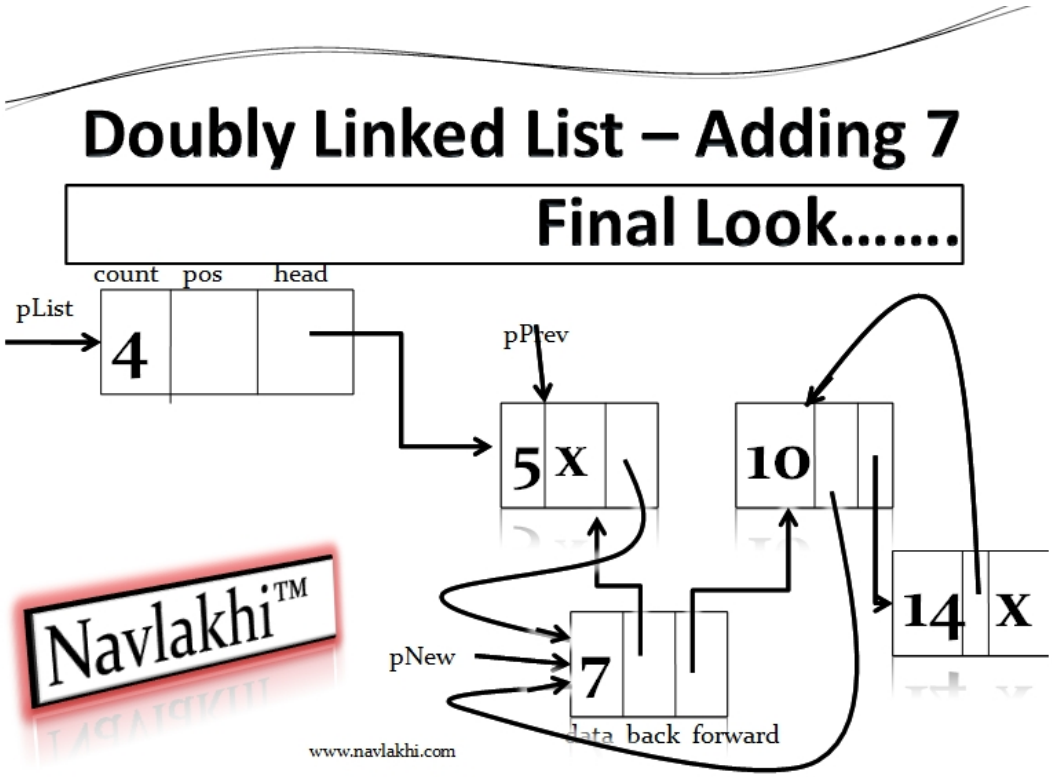

Doubly Linked List – Adding 7
pNew->back= pPrev
count pos head
pList
3
pPrev
5 X
10
14 X
pNew
7
data back forward
Navlakhi™
www.navlakhi.com

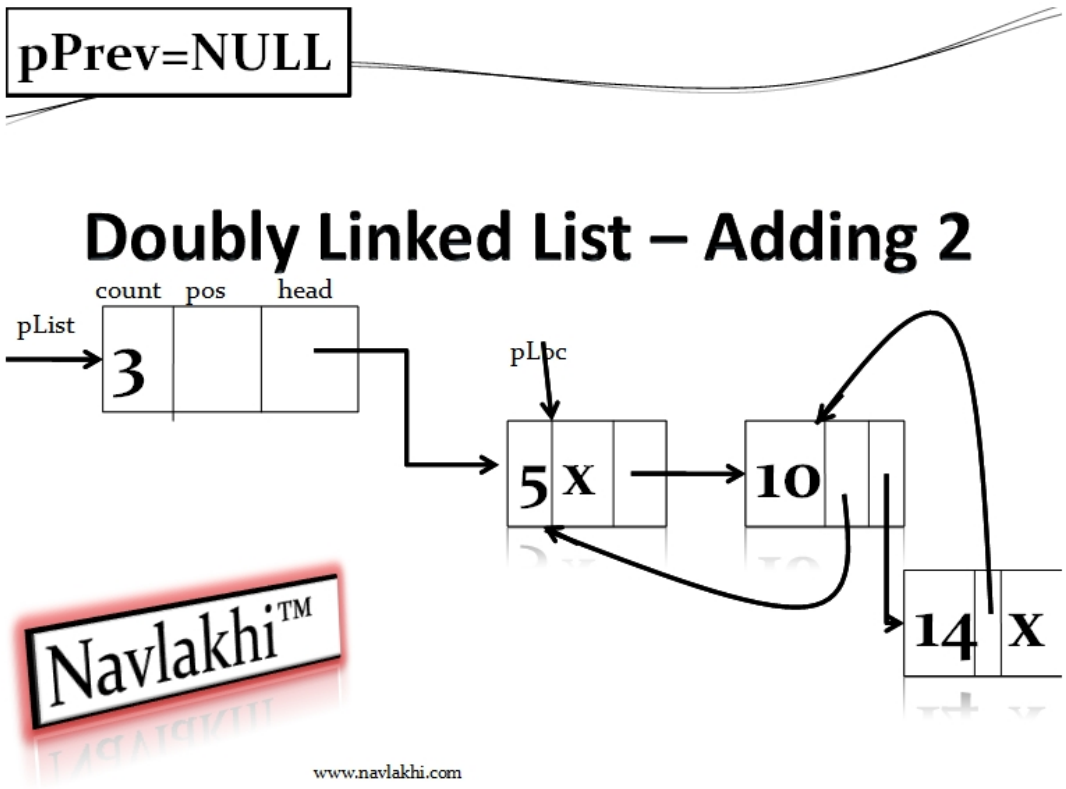

Doubly Linked List – Adding 7
If pPrev->forward exists
pPrev->forward->back= pNew
count pos head
pList
3
pPrev
5 X
?
10
14 X
pNew
7
data back forward
Navlakhi™
www.navlakhi.com

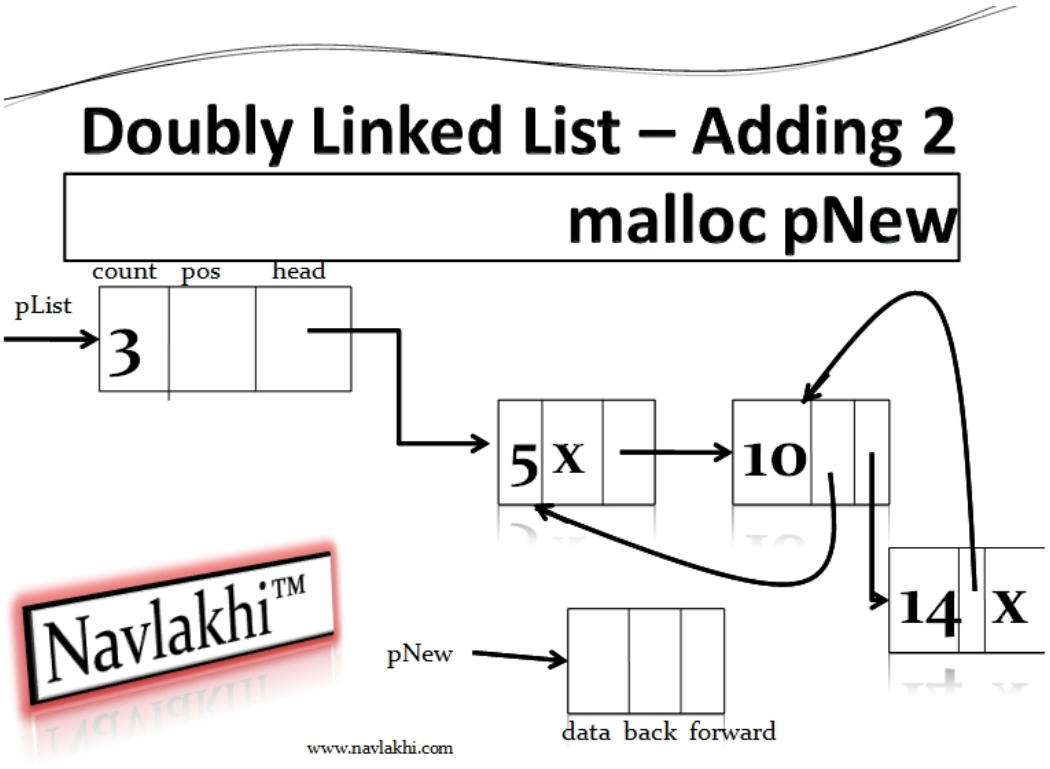

Doubly Linked List – Adding 7
If pPrev->forward exists
pPrev->forward->back= pNew
count pos head
pList
3
pPrev
5 X
?
10
14 X
pNew
7
data back forward
Navlakhi™
www.navlakhi.com

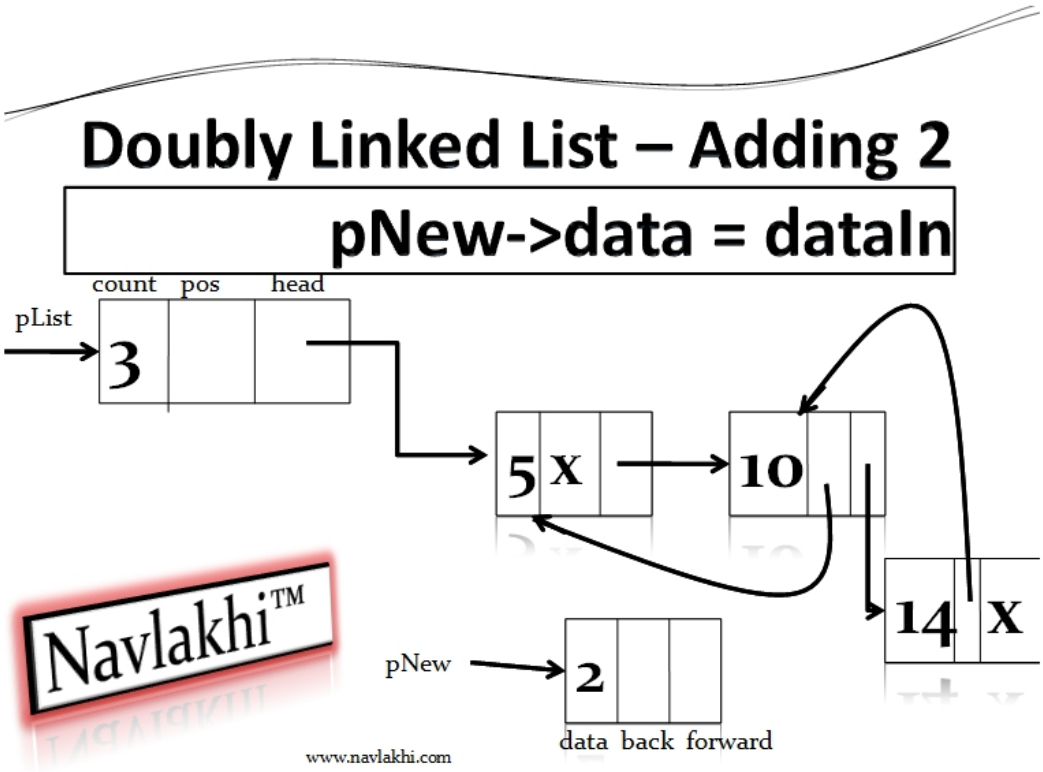

Doubly Linked List – Adding 7
pPrev->forward= pNew
count pos head
pList
3
pPrev
5 X
10
14 X
pNew
7
data back forward
Navlakhi™
www.navlakhi.com

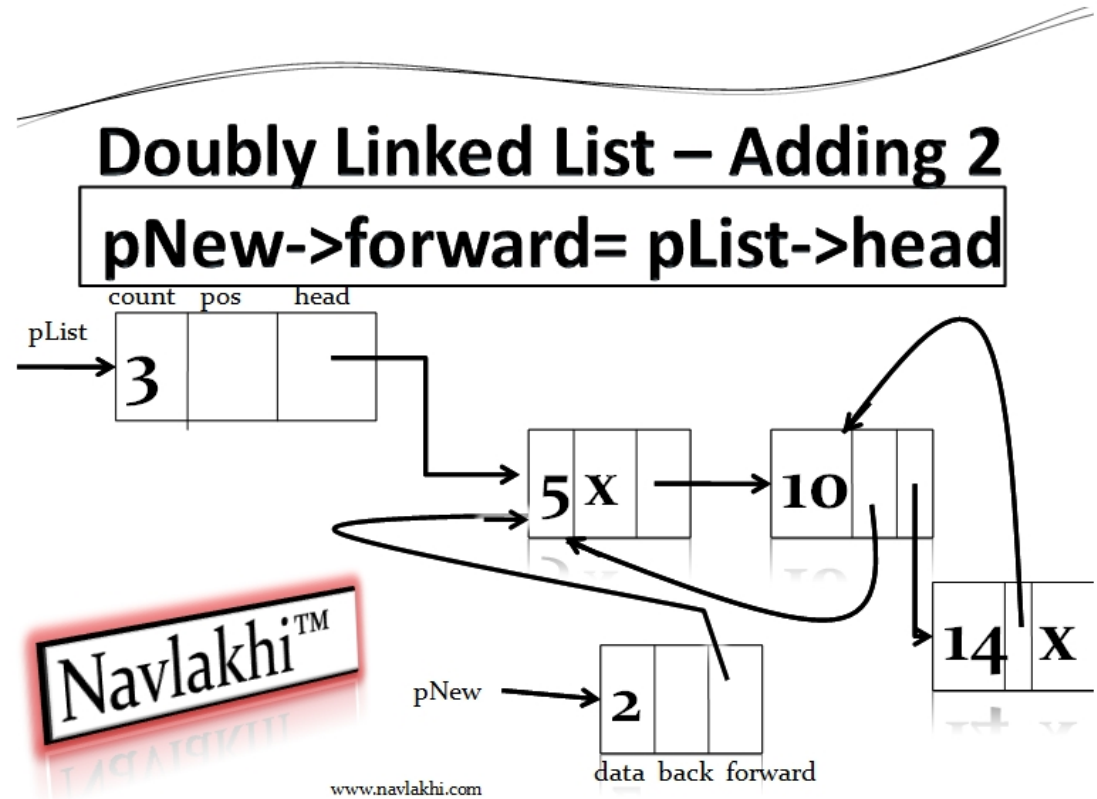

Doubly Linked List – Adding 7
pList->count++
count pos head
pList
4
pPrev
5 X
10
14 X
pNew
7
data back forward
Navlakhi™
www.navlakhi.com

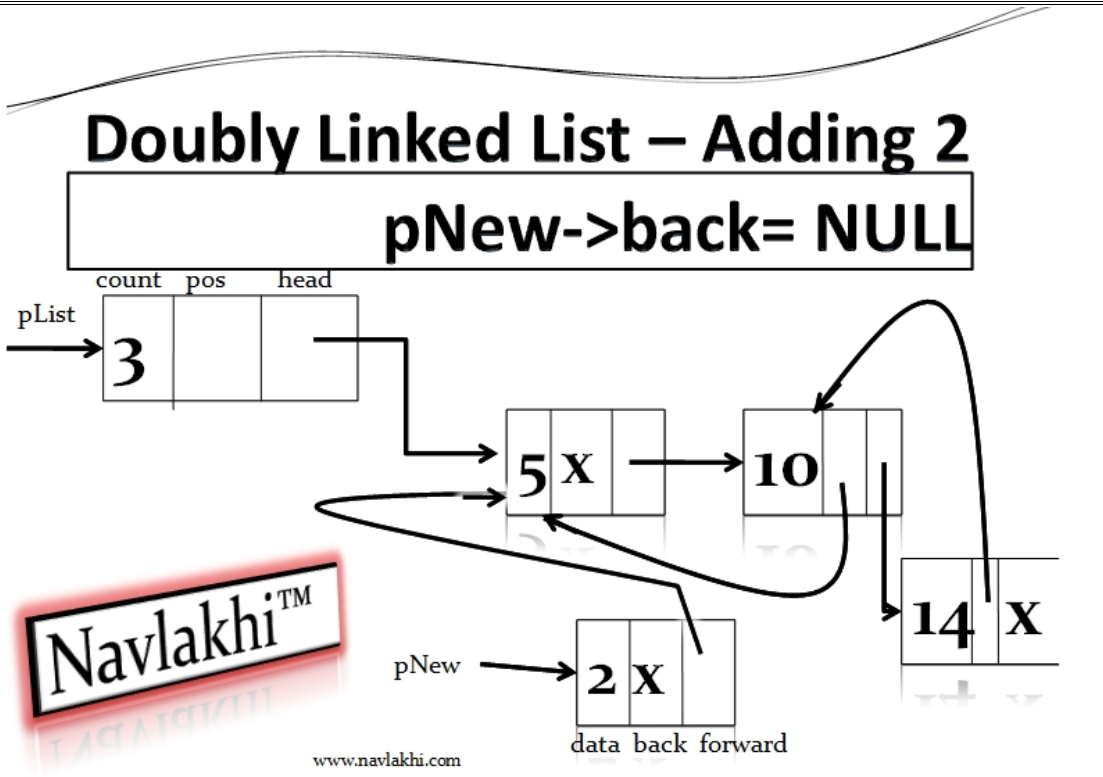

Doubly Linked List – Adding 7
Final Look.....
count pos head
pList
4
pPrev
5 X
10
14 X
pNew
7
data back forward
Navlakhi™
www.navlakhi.com

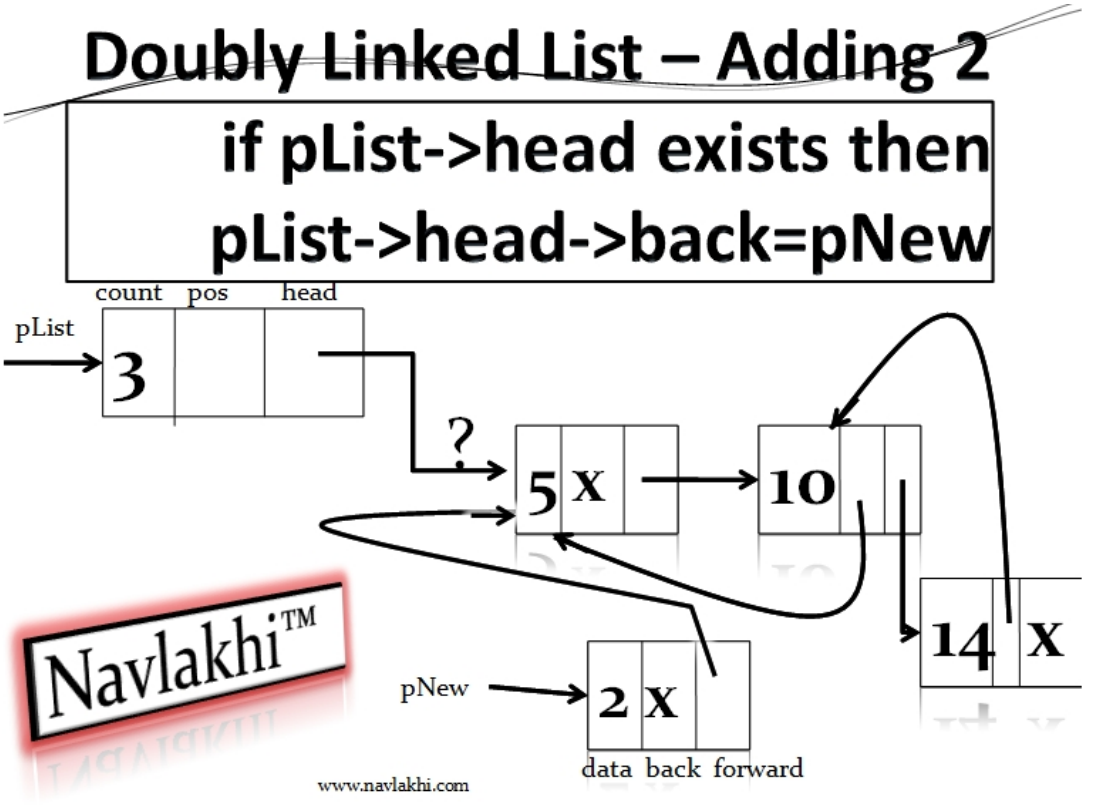

pPrev=NULL
Doubly Linked List – Adding 2
count pos head
pList
3
pLoc
5 X
10
14 X
Navlakhi™
www.navlakhi.com

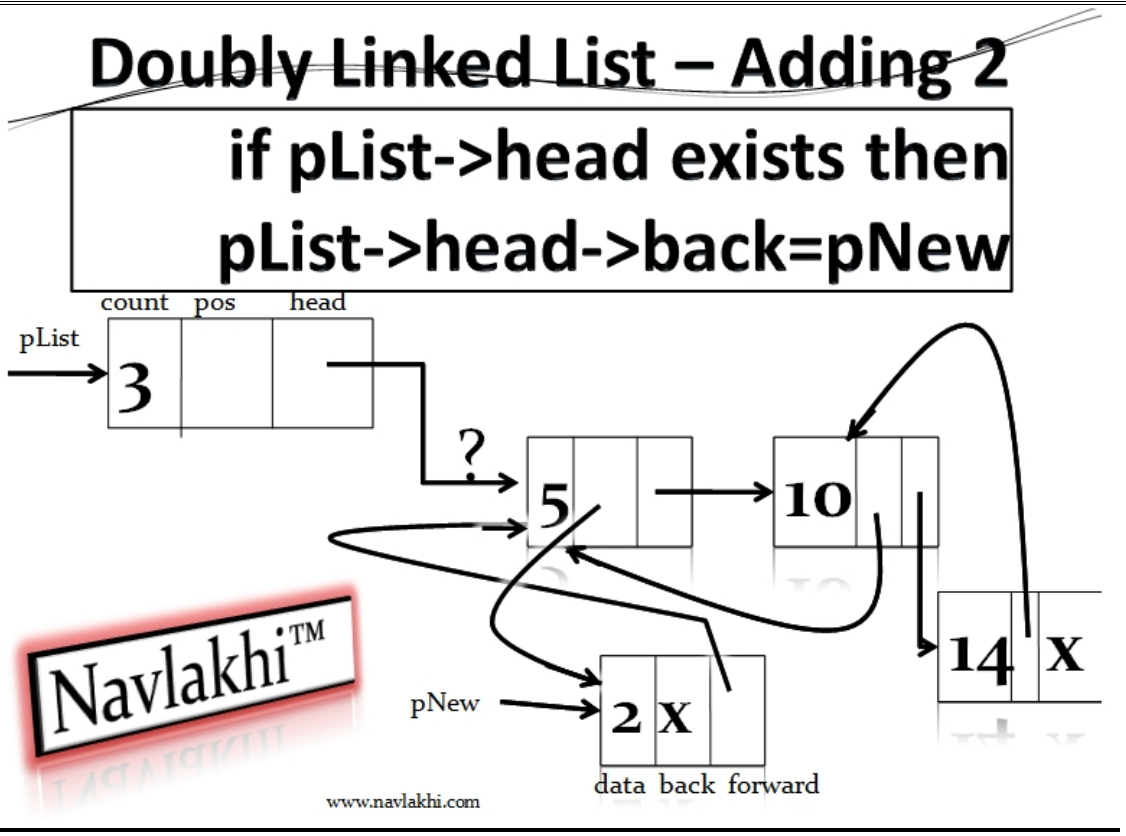

Doubly Linked List – Adding 2
malloc pNew
count pos head
pList
3
5 X
10
14 X
Navlakhi™
pNew
data back forward
www.navlakhi.com

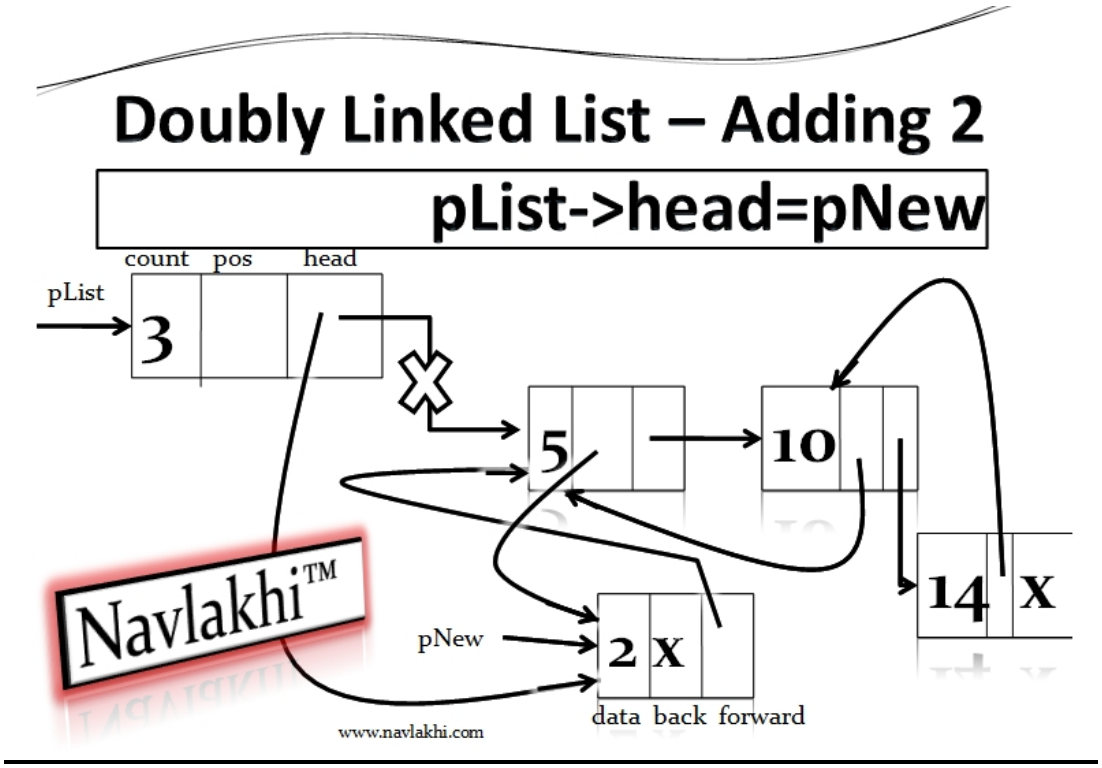

Doubly Linked List – Adding 2
pNew->data = dataIn
count pos head
pList
3
5 X
10
14 X
pNew
2
data back forward
Navlakhi™
www.navlakhi.com

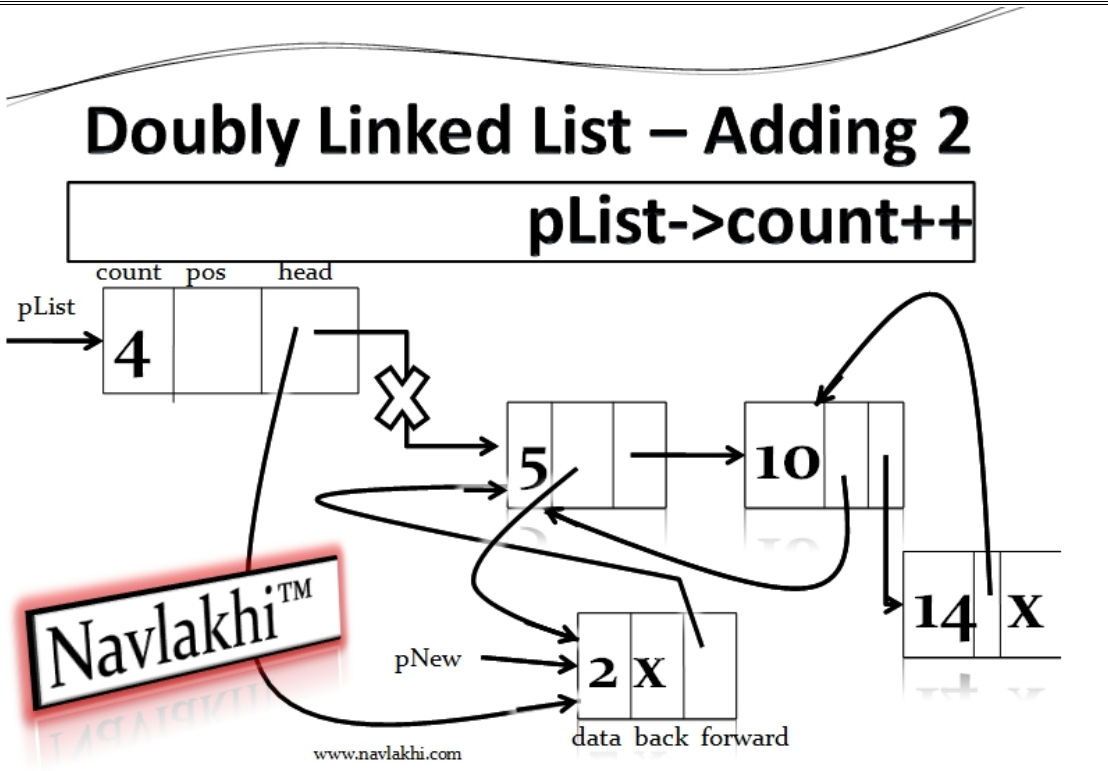

Doubly Linked List – Adding 2
pNew->forward= pList->head
count pos head
pList
3
5 X
10
14 X
pNew
2
data back forward
Navlakhi™
www.navlakhi.com

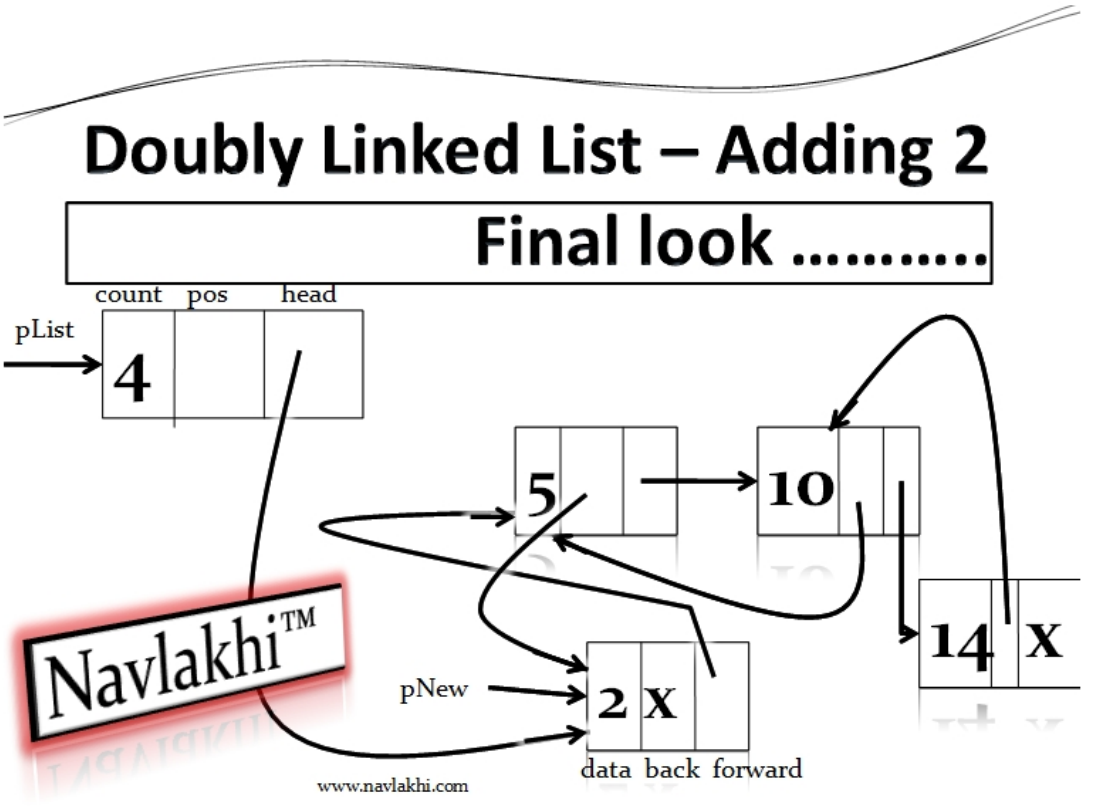

Doubly Linked List – Adding 2
pNew->back= NULL
count pos head
pList
3
5 X
10
14 X
pNew
2 X
data back forward
Navlakhi™
www.navlakhi.com


Doubly Linked List – Adding 2
if pList->head exists then
pList->head->back=pNew
count pos head
pList
3
?
5 X
10
14 X
pNew
2 X
data back forward
Navlakhi™
www.navlakhi.com

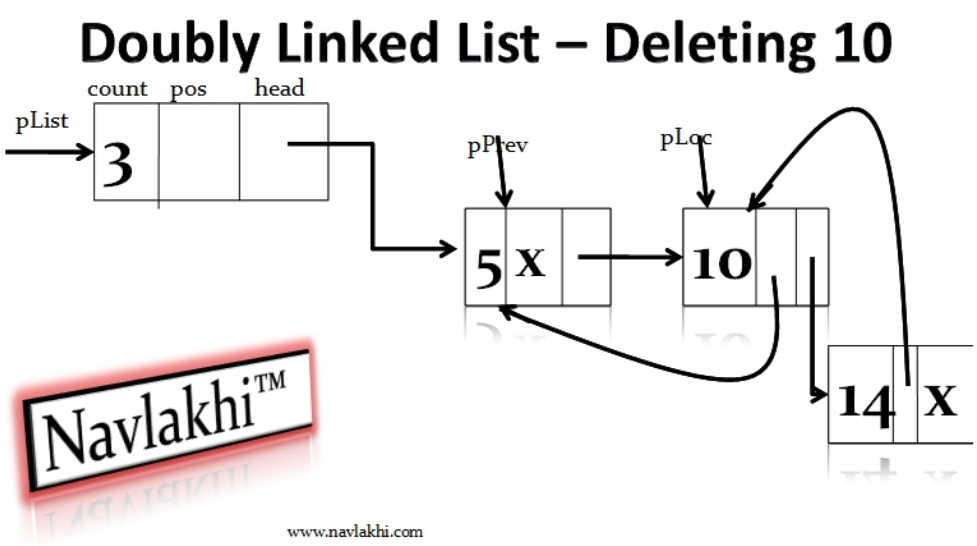

Doubly Linked List – Adding 2
if pList->head exists then
pList->head->back=pNew
count pos head
pList
3
?
5
10
14 X
pNew
2 X
data back forward
Navlakhi™
www.navlakhi.com

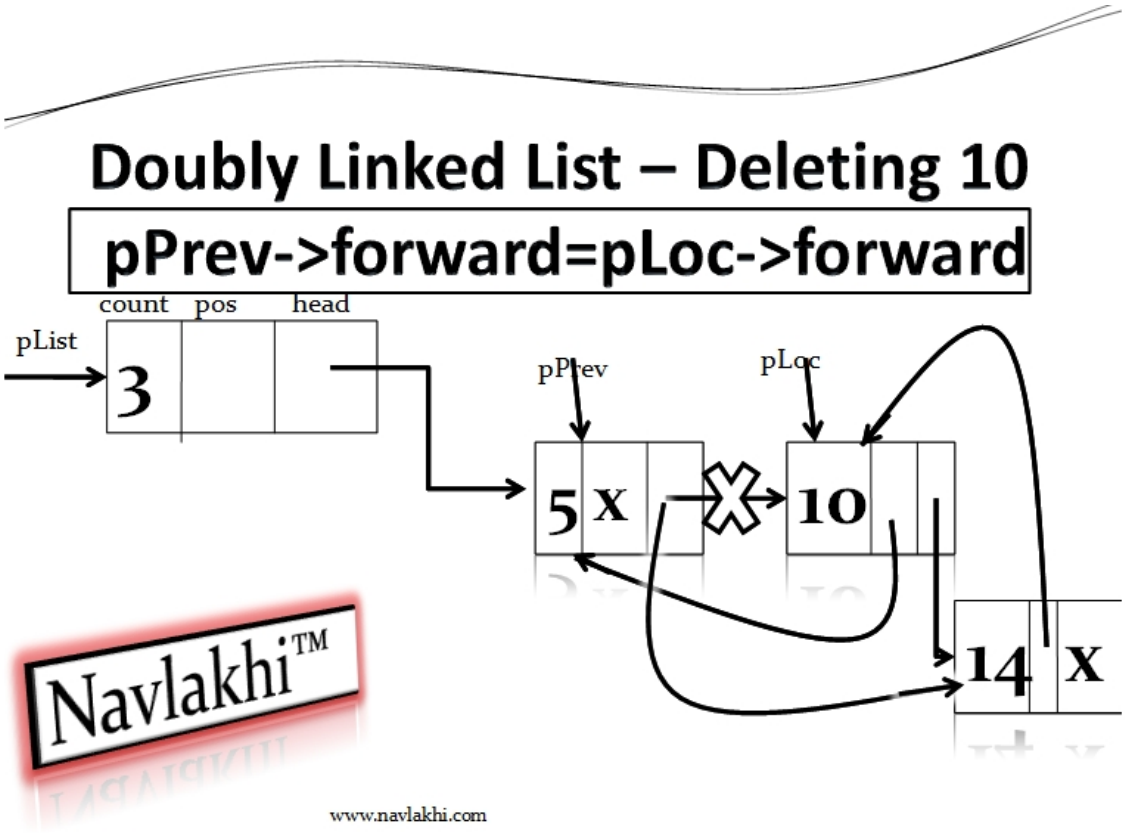

Doubly Linked List – Adding 2
pList->head=pNew
count pos head
pList
3
5
10
14 X
pNew
2 X
data back forward
Navlakhi™
www.navlakhi.com

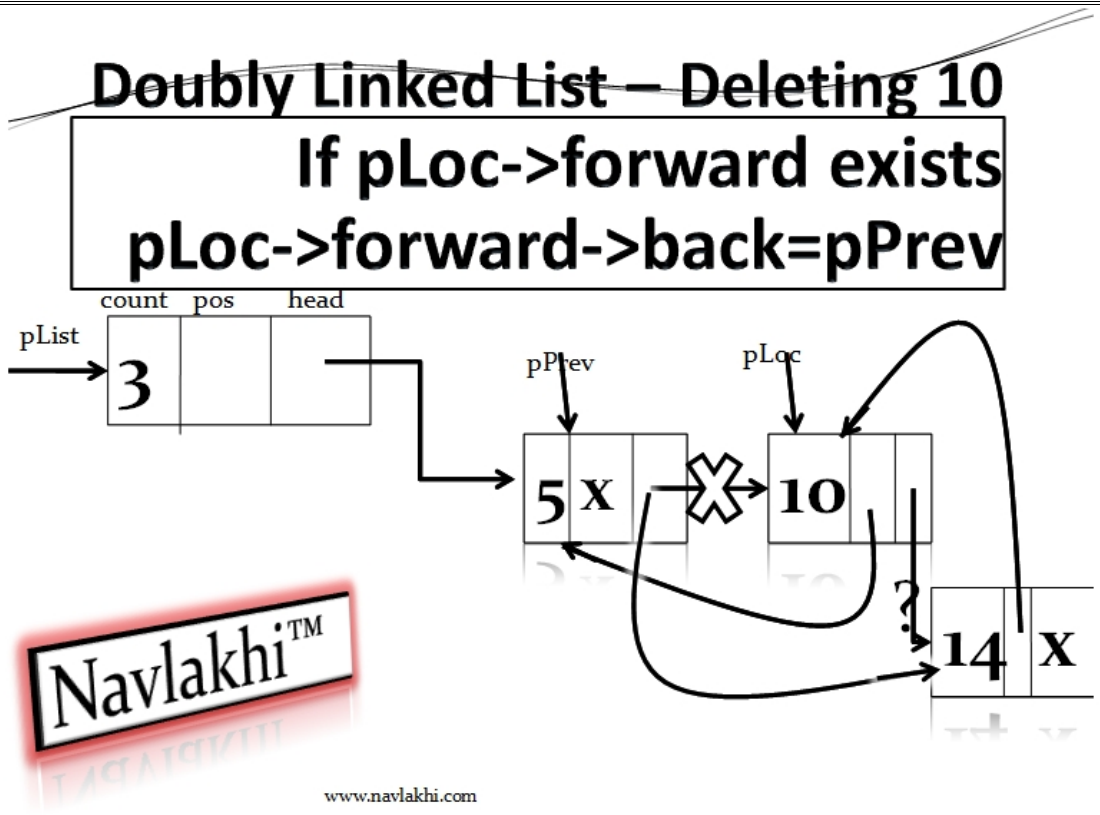

Doubly Linked List – Adding 2
pList->count++
count pos head
pList
4
5
10
14 X
pNew
2 X
data back forward
Navlakhi™
www.navlakhi.com

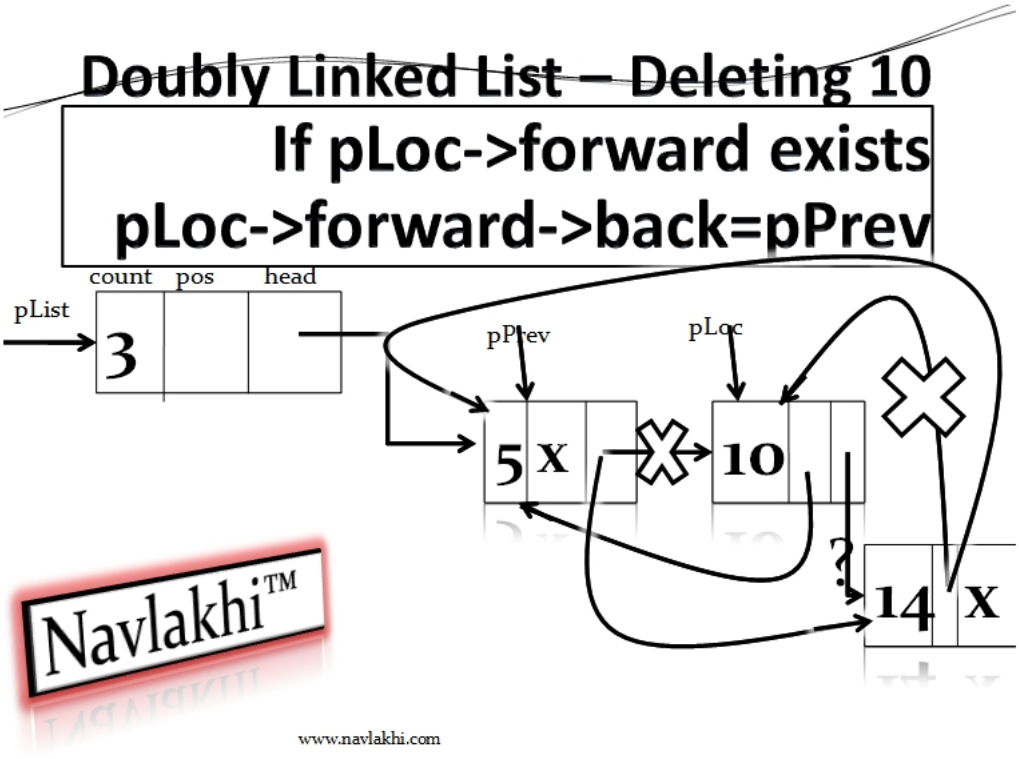

Doubly Linked List – Adding 2
Final look .....
count pos head
pList
4
5
10
14 X
pNew
2 X
data back forward
Navlakhi™
www.navlakhi.com

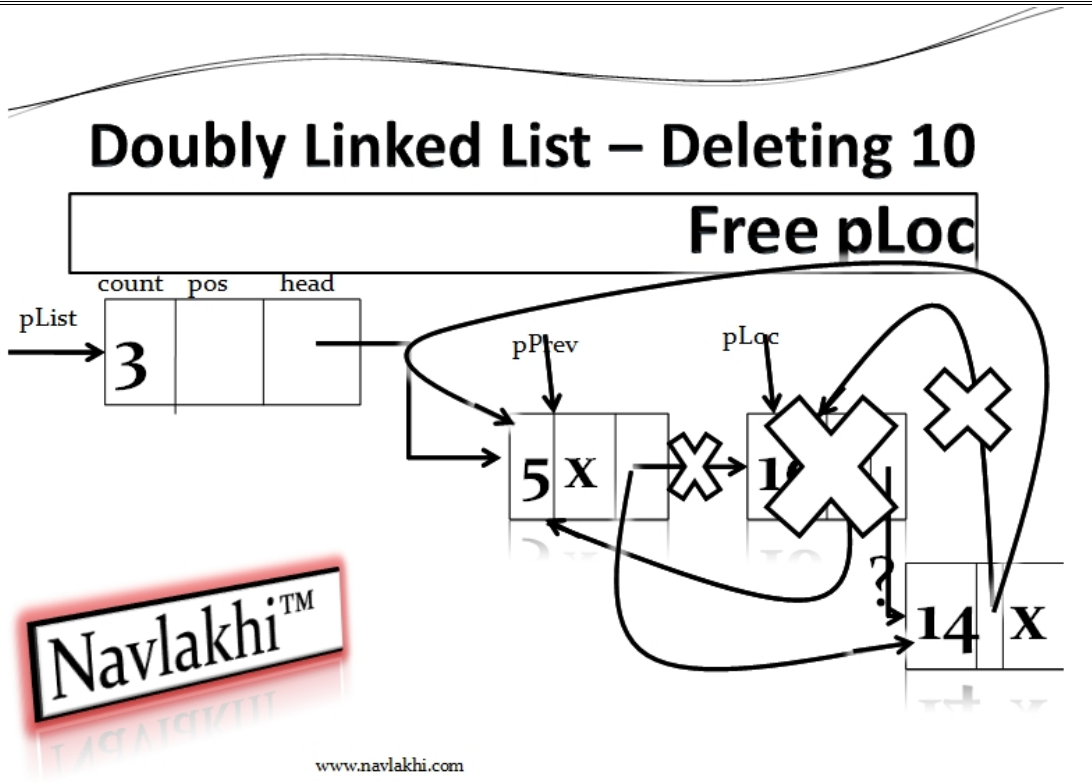
DELETING A NODE Case 1: pPrev!=NULL (Not deleting the 1st Node)

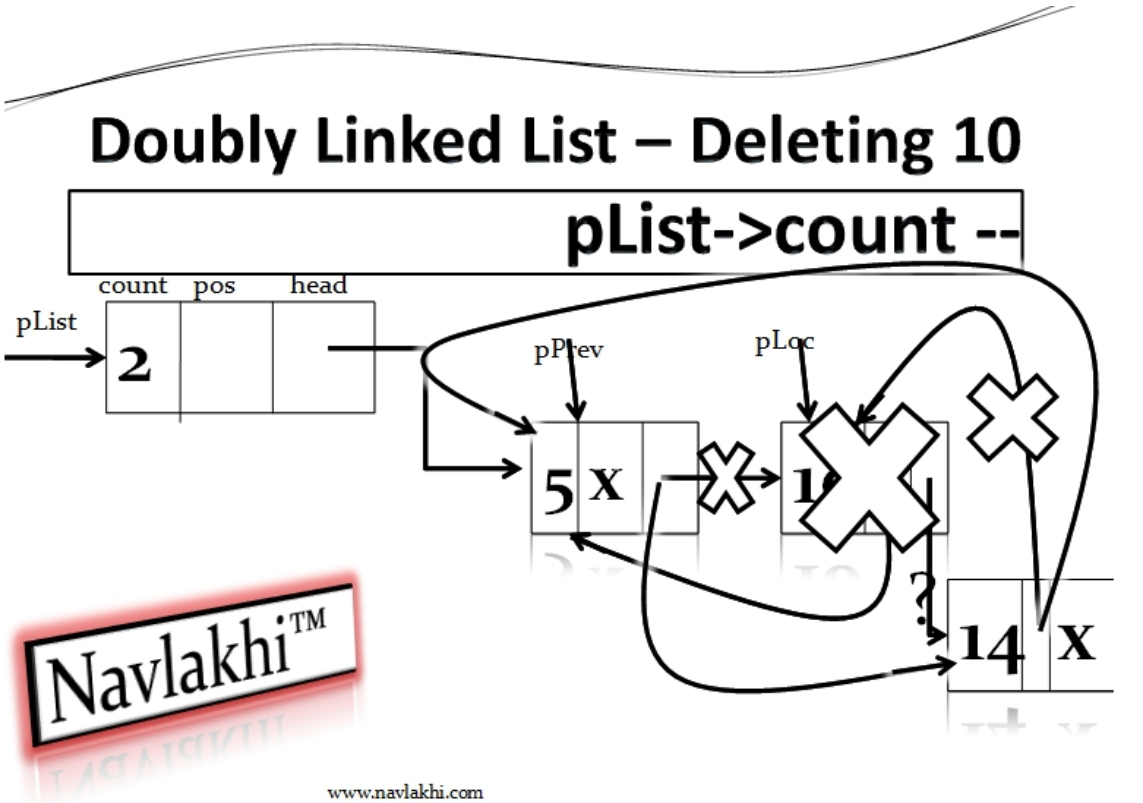


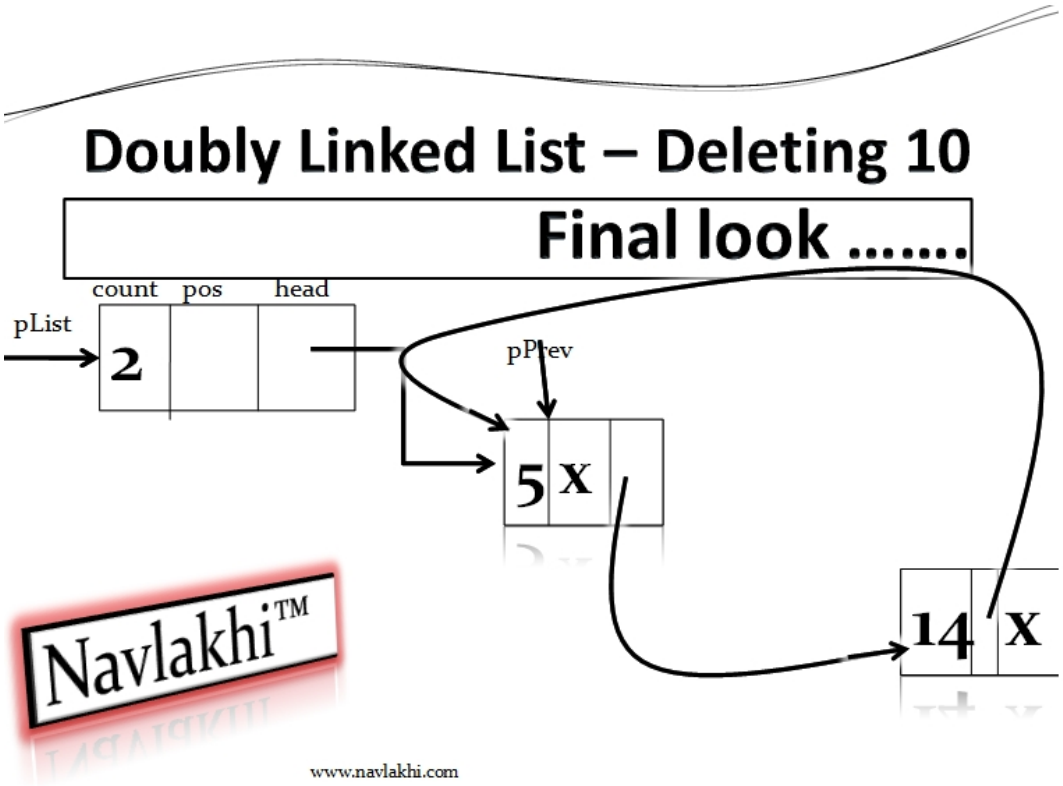


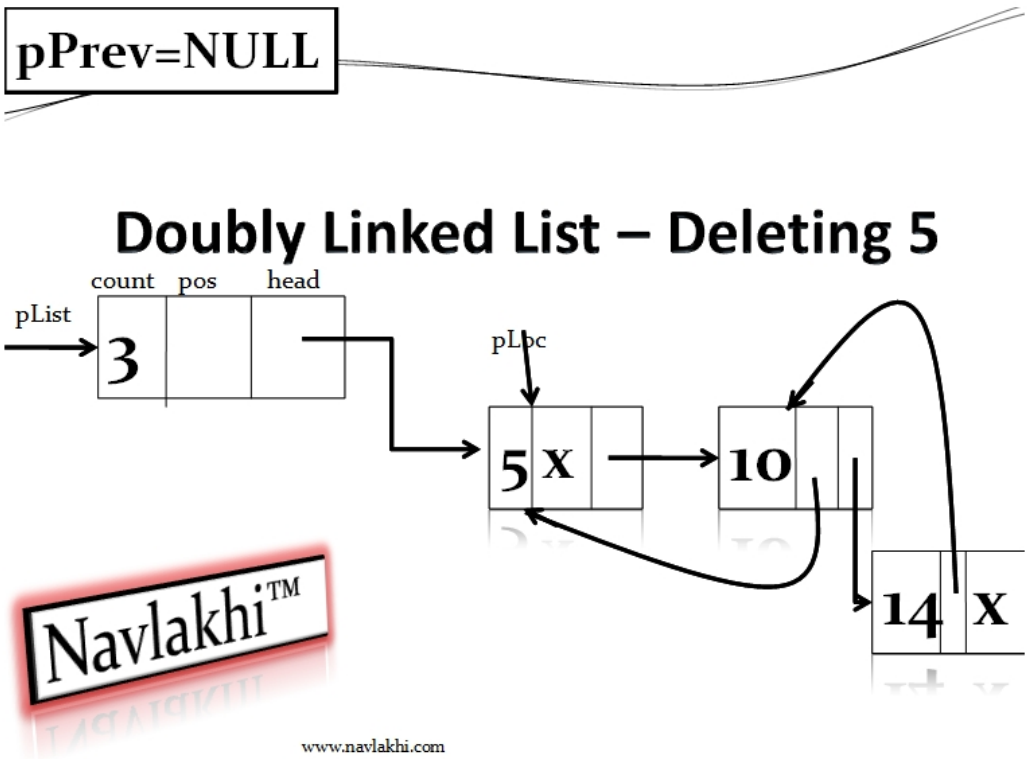

Doubly Linked List – Deleting 10
If pLoc->forward exists
pLoc->forward->back=pPrev
count
pos
head
pList
3
pPrev
pLoc
5
X
10
?
14
X
Navlakhi™
www.navlakhi.com

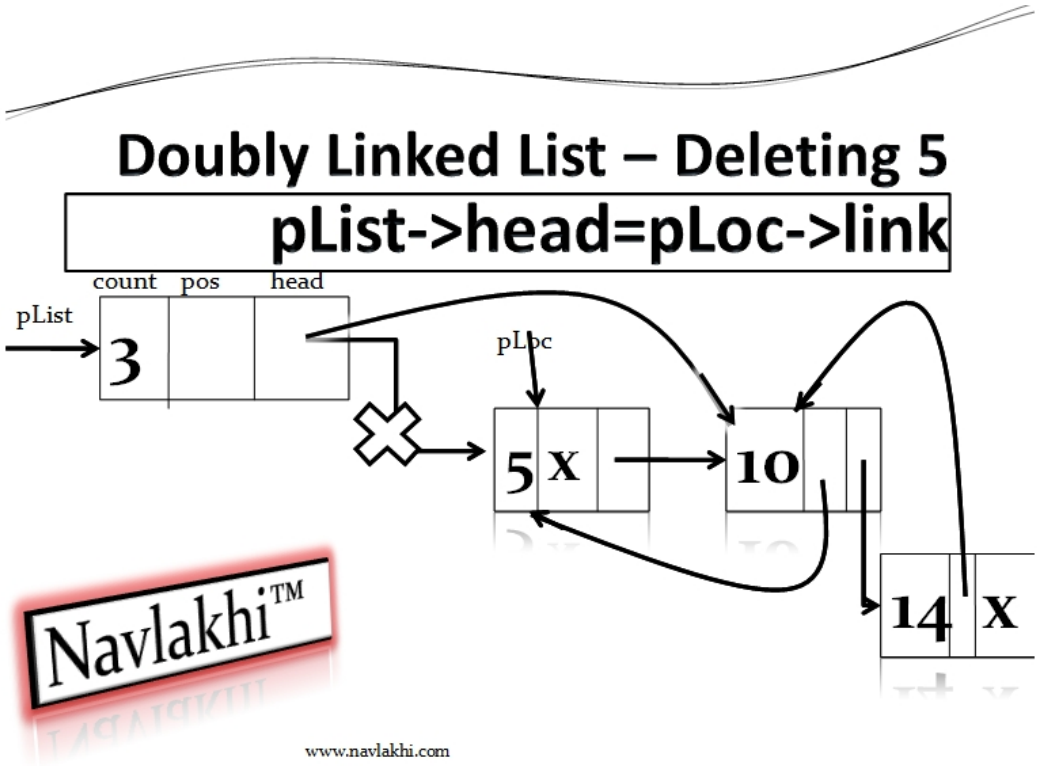

Doubly Linked List – Deleting 10
If pLoc->forward exists
pLoc->forward->back=pPrev
count
pos
head
pList
3
pPrev
pLoc
5
X
10
?
14
X
Navlakhi™
www.navlakhi.com

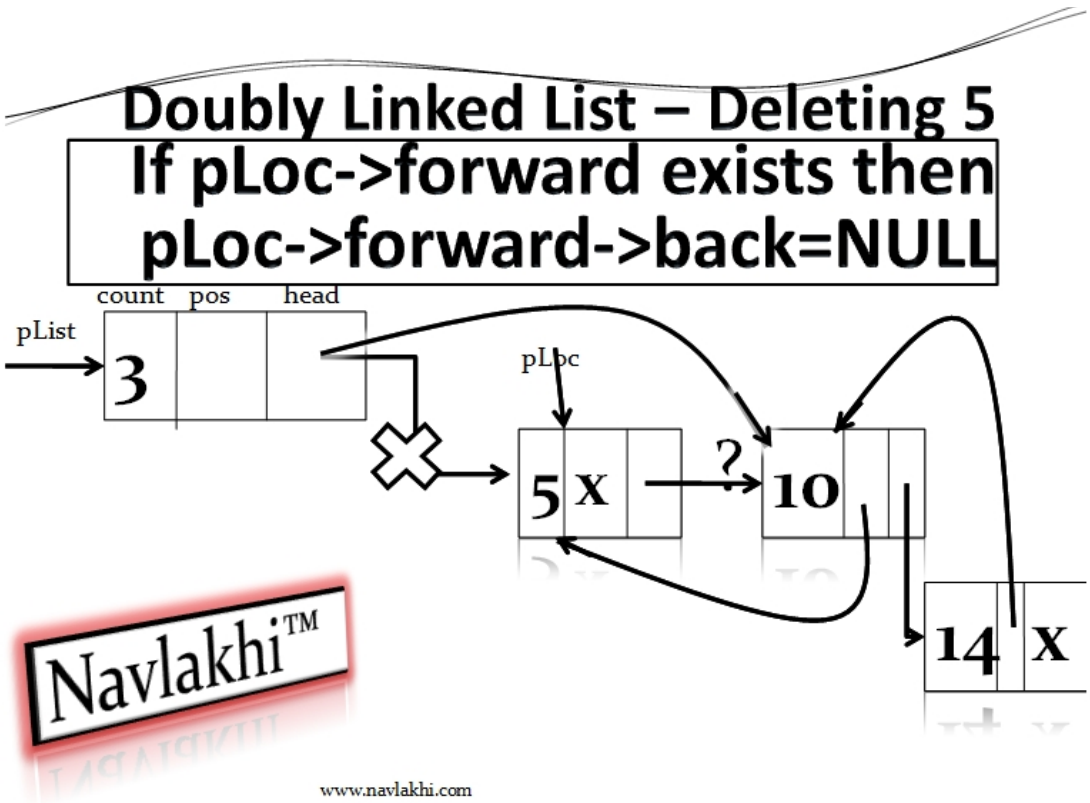

Doubly Linked List – Deleting 10
Free pLoc
count pos head
pList
3
pPrev
pLoc
5 X
14 X
Navlakhi™
www.navlakhi.com

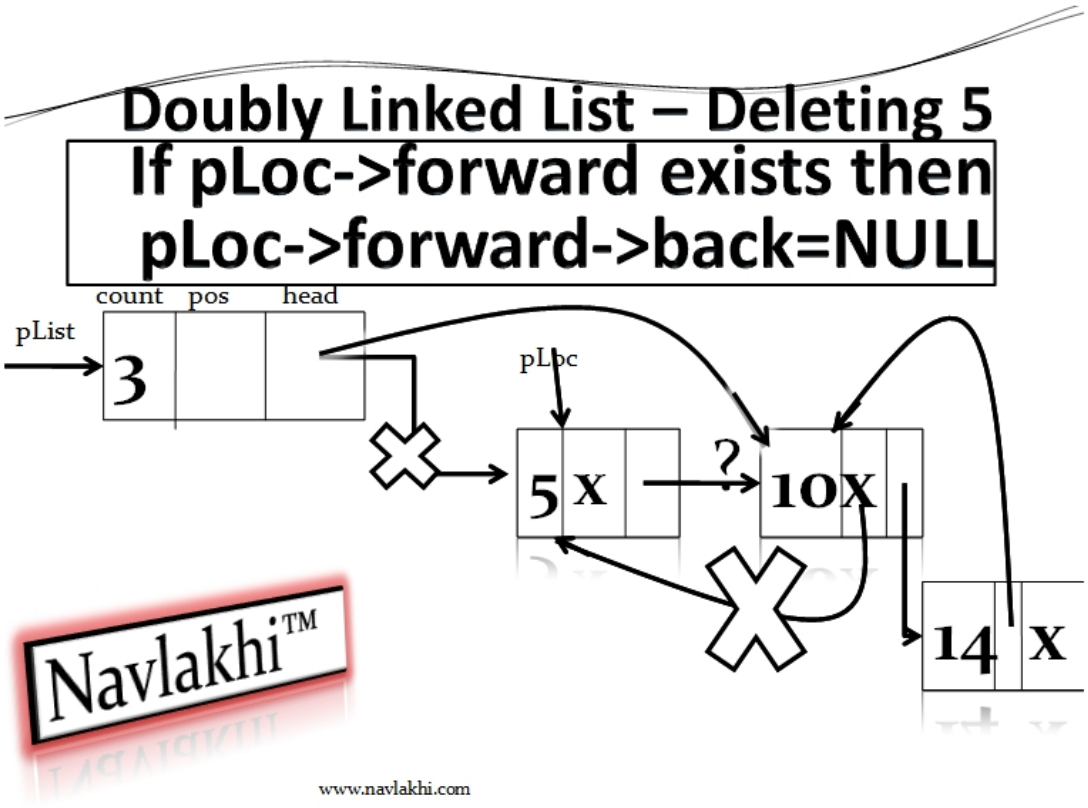

Doubly Linked List – Deleting 10
pList->count --
count pos head
pList
2
pPrev
pLoc
5 X
14 X
Navlakhi™
www.navlakhi.com

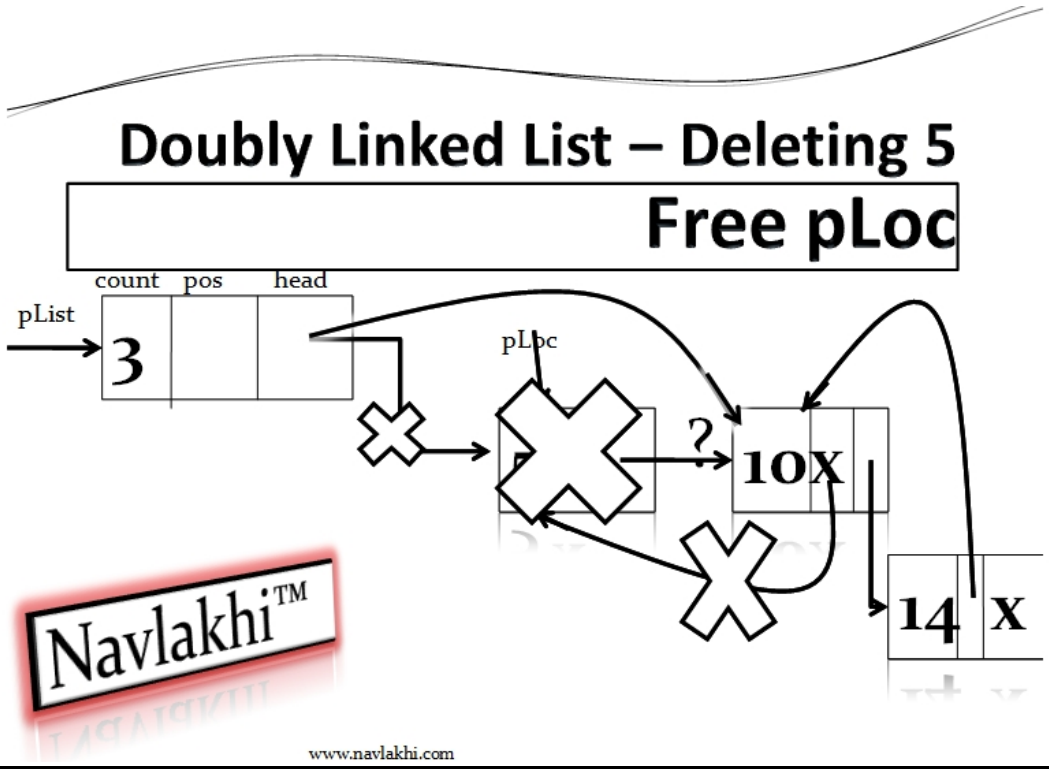

Doubly Linked List – Deleting 10
Final look .....
count pos head
pList
2
pPrev
5 X
14 X
Navlakhi™
www.navlakhi.com

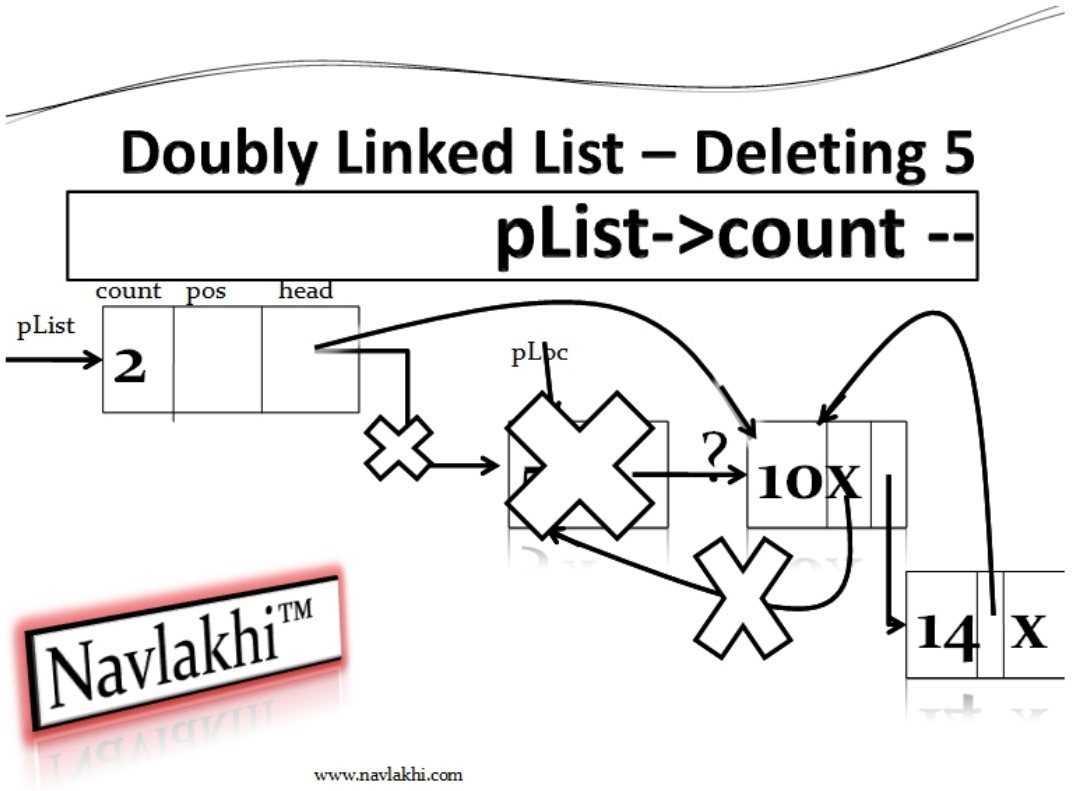

pPrev=NULL
Doubly Linked List – Deleting 5
count pos head
pList
3
pLoc
5 X
10
14 X
Navlakhi™
www.navlakhi.com


Doubly Linked List – Deleting 5
pList->head=pLoc->link
count pos head
pList
3
pLoc
5 X
10
14 X
Navlakhi™
www.navlakhi.com


Doubly Linked List – Deleting 5
If pLoc->forward exists then
pLoc->forward->back=NULL
count pos head
pList
3
pLoc
5 X
?
10
14 X
Navlakhi™
www.navlakhi.com


Doubly Linked List – Deleting 5
If pLoc->forward exists then
pLoc->forward->back=NULL
count
pos
head
pList
3
pLoc
5
X
?
10
X
14
X
Navlakhi™
www.navlakhi.com


Doubly Linked List – Deleting 5
Free pLoc
count
pos
head
pList
3
pLoc
?
10
X
14
X
Navlakhi™
www.navlakhi.com


Doubly Linked List – Deleting 5
pList->count --
count
pos
head
pList
2
pLoc
?
10X
14
X
Navlakhi™
www.navlakhi.com

Program

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

struct node
{
int data;
struct node *forward;
struct node *back;
};

struct headnode
{
int count;
struct node *pos;
struct node *head;
}*pList;

struct node *pPrev,*pLoc
```

```
void printList( )
{
        int i;

        pList->pos=pList->head;
        for (i=1;i<=pList->count;i++)
        {
                printf("%d\t",pList->pos->data);
                pList->pos=pList->pos->forward;
        }
}
```

```
int searchNode(int target)
{

pPrev=NULL;

pLoc=pList->head;

while(pLoc!=NULL && target>pLoc->data)
{
        pPrev=pLoc;
        pLoc=pLoc->forward;
}


if (pLoc==NULL)
        return 0; /*Not found*/
else
        if (target == pLoc->data) return 1; /*FOUND*/
else
        return 0;

}
```

```
void deleteNode( )
{
if (pPrev!=NULL)
{
        pPrev->forward=pLoc->forward;

        if (pLoc->forward!=NULL)
                pLoc->forward->back=pPrev;
}
else
{
        pList->head=pLoc->forward;
        if (pLoc->forward!=NULL)
                pLoc->forward->back=NULL;

}

pList->count =pList->count - 1;
free(pLoc);
}
```

```
void removeNode(int key)
{
int found;

found=searchNode(key);

if (found)
        deleteNode( );
else
        printf("Error: No matching data found\n");
}
```

```
int insertNode( int dataIn)
{
struct node *pNew;
pNew = (struct node *) malloc(sizeof(struct node));
if (pNew != NULL)
{
        pNew->data=dataIn;
        if (pPrev!=NULL)
        {
                pNew->back=pPrev;
                pNew->forward=pPrev->forward;
                if (pPrev->forward!=NULL)/*OR if (pNew->forward!=NULL)*/
                        pPrev->forward->back=pNew;

                pPrev->forward=pNew;
        }
        else
        {
                pNew->back=NULL;
                pNew->forward=pList->head;
                if (pNew->forward!=NULL) pList->head->back=pNew;
                pList->head=pNew;
        }
        pList->count +=1;
        return 1;
}
else    return 0;
}
```

```
void addNode( int dataIn)
{
int found,success;
found = searchNode(dataIn);
if (found)     printf("Data already inserted\n");
else
{
        success=insertNode(dataIn);
        if (success)  printf("Data Inserted Successfully\n");
        else          printf("Out of Memory.....\n");
}
}
```

```
int menu( )
{
int choice;
printf("\n\n********************\n\n");
printf(" ....   M E N U ...\n");
printf("1: Add new data\n");
printf("2: Delete data\n");
printf("3: Print List\n");
printf("4: Quit\n\n");
printf("********************\n\n");

printf("feed in your choice: ");
scanf("%d",&choice);

return choice;
}
```

```
void createList( )
{
pList = (struct headnode *)malloc(sizeof(struct headnode));
if (pList != NULL)
{
        pList->count=0;
        pList->head = NULL;
}
else
{
        printf("Insufficient memory...\n");
        exit(0);
}
}
```

```
void main( )
{
int choice;
int dataIn,deleteKey;
clrscr();
createList( );

do
{
        choice = menu();

        if (choice==1)
    {
                printf("Feed in the data: ");
                scanf("%d",&dataIn);
                addNode(dataIn);
    }
        else
        if (choice==2)
        {
                  printf("Enter key to be deleted: ");
                  scanf("%d",&deleteKey);
                  removeNode(deleteKey);
        }
        else
        if (choice==3)
            printList( );

} while(choice!=4);

}
```

HOME OF EDUCATION

Navlakhi®

www.navlakhi.com
Home of Education

DATA Structures@ Navlakhi's