UNIT : 4

LINKED LISTS

A linked list is a collection of data element called nodes in which the linear representation is given by links from one node to the next node.

We have studied that an array is a linear collection of data elements in which the element ~~in~~ which the are stored in consecutive memory locations. while declaring arrays, we have to specify the size of array, which will restrict the number of elements that the array can store. for example, if we declare an array as int mark[10], then the array can store a maximum of 10 data elements but not more than that. But what if we are not sure of the number of elements in advance? So, there must be a data structure that removes the restrictions on the maximum number of elements and the storage condition to write efficient programs.

Linked list is a data structure that is free from the aforementioned restrictions. A linked list does not store its elements in consecutive memory locations and the user can add any number of elements to it. However unlike an array, a linked list does not allow random access of data. Elements in a linked list can be accessed only in a

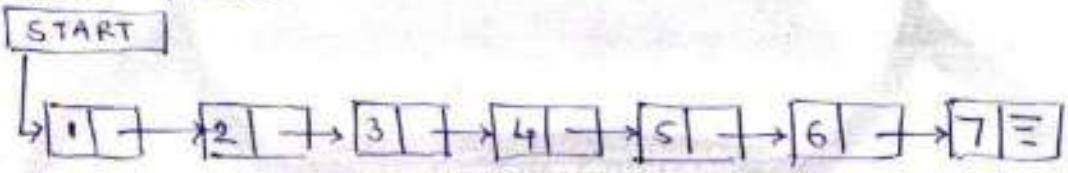
A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

Semester: 3 Subject: DSA Academic Year: 2017-2018

sequential manner But like an array, insertions and deletions can be done at any point in the list In a constant time.

Basic Concept of Linked List

A linked list, in simple terms, is a linear collection of data elements. These data elements are called nodes. Linked List is a data structure which in turn can be used to implement other data structures. Thus it acts as a building block to implement data structure such as stacks, queues and their variations. A linked list can be perceived as a train or a sequence of nodes in which each node contains one or more data fields and a pointer to the next node.



Simple linked list

we can see a simple linked list in which every node contains ~~data~~ two parts, an integer and a pointer to the next node. The left part of the node which contains data may include a simple data type, an array or a structure. The right part of the node contains a pointer to the next node (or address of the next node in sequence). The last node will have no next node connected to it so it will store a special value called NULL.

Hence, a NULL pointer denotes the end of the list.

A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

Semester: 3 Subject: DSA Academic Year: 2017-2018

Linked lists contain a pointer variable START that stores the address of the first node in the list. We can traverse the entire list using START which contains the address of the first node; the next part of the first node in turn stores the address of the first node; the next part of the first node in turn stores the address of its succeeding node. Using this technique the individual node of the list will form a chain of nodes. If START = NULL, then the linked list is empty and contains no nodes.

In C, we can implement a linked list using the following code:

```
struct node
{
    int data;
    struct node * next;
};
```

Linked Lists verses Arrays

Both arrays and linked lists are a linear collection of data elements. But unlike an array, a linked list does not store its nodes in consecutive memory location. Another point of difference between an array and a linked link list is that a linked list does not allow random access of data. Nodes in a linked list can be accessed only in a sequential manner. But like an array insertions and deletions can be done at any point in the list in a constant time.

Semester: 3 Subject: DSA Academic Year: 2017-2018

Another advantage of linked list over an array is that we can add any number of elements in the list. This is not possible in case of an array.

Thus linked lists provide an efficient way of storing related data and performing basic operations such as insertion, deletion and updation of information at the cost of extra space required for storing the address of next nodes.

Memory Allocation and De-allocation for a Linked List

We have seen how a linked list is represented in the memory. If we want to add a node to an already existing linked list in the memory, we first find free space in the memory and then use it to store the information for example, Consider the linked list shown in following figure The linked list contains the roll number of students, marks obtained by them in Biology and finally a next field which stores the address of the next node in sequence Now, if a new student joins the class and is asked to appear for the same test that the other students had taken, then the new student's marks should also be recorded in the linked list. For this purpose, we find a free space and store the information there. In following fig. the grey shaded portion shows free space, and thus we have 4 memory

A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

Semester: 3 Subject: DSA Academic Year: 2017-2018

locations available. We can use any one of them to store our data. This is illustrated in following fig. We can use any one of them to store out data.

Now, the question is which part of the memory is available and which part is occupied? When we delete a node from a linked list, then who changes the status of the memory occupied by it from occupied to available? The answer is the operating system

START → 1 → BIOLOGY

	Roll No	Marks	Next
1	S01	78	2
2	S02	84	3
3	S03	45	5
4			
5	S04	98	7
6	S		
7	S05	55	8
8	S06	34	10
9			
10	S07	90	11
11	S08	87	12
12	S09	86	13
13	S10	67	15
14			
15	S11	56	-1

(a)

START → 1 → BIOLOGY

	Roll No	Marks	Next
1	S01	78	2
2	S02	84	3
3	S03	45	5
4	S12	75	-1
5	S04	98	7
6			
7	S05	55	8
8	S06	34	10
9			
10	S07	90	11
11	S08	87	12
12	S09	86	13
13	S10	67	15
14			
15	S11	56	4

(b)

(a) Student's linked list and (b) linked list after the insertion of new student's record.

Padmabhushan Vasantdada Patil's
A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

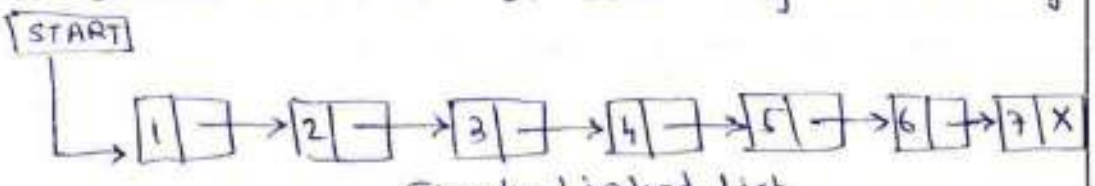
Semester: 3 Subject: DSA Academic Year: 2017-2018

When we delete a particular node from an existing linked list or delete the entire linked list, the space occupied by it must be given back to the free pool so that the memory can be reused by some other program that needs memory space.

The operating system does this task of adding freed memory to the free pool. The operating system will perform this operation whenever it finds the CPU idle or whenever the programs are falling short of memory space. The operating system scans through all the memory cells and marks those cells that are being used by some program. This process is called garbage collection.

SINGLY LINKED LISTS

A singly linked list is the simplest type of linked list in which every node contains some data and a pointer to the next node of the same data type. By saying that the node contains a pointer to the next node, we mean that the node stores the address of the next node sequence. A singly linked list allows traversal of data only in one way



Singly Linked list

A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

Semester: 3 Subject: DSA Academic Year: 2017-2018

Traversing a Linked List

Traversing a linked list means accessing the nodes of the list in order to perform some processing on them. Remember a linked list always contains a pointer variable START which stores the address of the first node of the list. End of the list is marked by storing NULL or -1 in the NEXT field of the last node. For traversing the linked list, we also make use of another pointer variable PTR which points to the node that is currently being accessed. The algorithm to traverse a linked list is shown in fig.

Step 1 : [INITIALIZE] SET PTR = START
Step 2 : Repeat Steps 3 and 4 while PTR != NULL
Step 3 : SET PTR = PTR → NEXT
[END OF LOOP]
Step 4
Step 3 : Apply Process to PTR → DATA
Step 4 : SET PTR = PTR → NEXT
[END OF LOOP]
Step 5 : EXIT

Searching for a Value in a Linked List

Searching a linked list means to find a particular element in the linked list. As already discussed, a linked list ~~contains~~ consists of nodes which are divided into two parts, the information part and the next part. So searching means finding

A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

Semester: 3 Subject: DSA Academic Year: 17-18

whether a given value is present in the information part and the next part. So searching means finding whether of the node or not. If it is present, the algorithm returns the address of the node that contains the value. The following fig shows the algorithm to search a linked list

```
Step 1 : [INITIALIZE] SET PTR = START
Step 2 : Repeat Step 3 while PTR != NULL
Step 3 : IF VAL = PTR → DATA
             SET POS = PTR
             Go to Step 5
         ELSE
             SET PTR = PTR → NEXT
         [END OF IF]
     [END OF LOOP]
Step 4 : SET POS = NULL
Step 5 : EXIT
```

Inserting a New Node in a Linked List

In this section, we will see how a new node is added into an already existing linked list. We will take four cases and then see how insertion is done in each case.

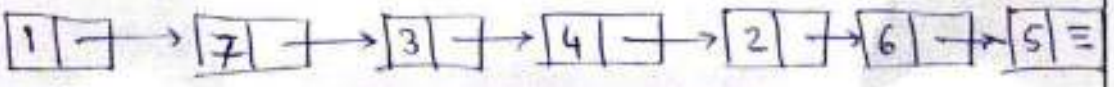
Case 1 : The new node is inserted at the beginning
Case 2 : The new node is inserted at the end
Case 3 : The new node is inserted after a given node
Case 4 : The new node is inserted before a given node.

A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

Semester: 3 Subject: DSA Academic Year: 17-18

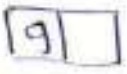
Inserting a Node at the Beginning of a linked List

Consider the linked list shown as below Suppose we want to add a new node with data 9 and add it as the first node of the list. Then the following changes will be done in the linked list.

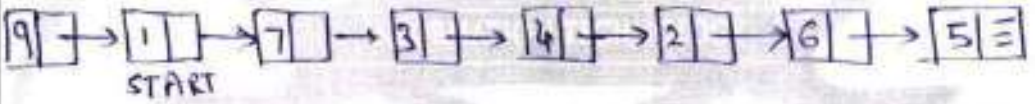


START

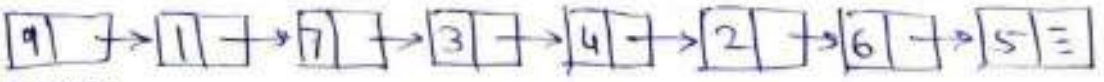
Allocate memory for the new node and initialize its DATA part to 9



Add the new node as the first node of the list by making the next part of the new node contain the address of START



Now make START to point to the first node of the list



Inserting an element at the beginning of a linked List.

A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

Semester: 3 Subject: DSA Academic Year: 17-18

```
step 1 : IF AVAIL = NULL
         Write OVERFLOW
         Go to Step 7
     [ END OF IF ]
step 2 : SET NEW_NODE = AVAIL
step 3 : SET NEW-NODE = AVAIL → NEXT
step 4 : SET NEW_NODE → DATA = VAL
step 5 : SET NEW_NODE → NEXT = START
step 6 : SET START = NEW_NODE
step 7 : EXIT
```

In step 1 we first check whether memory is available for the new node. If the free memory has exhausted, then an OVERFLOW message is printed.

Inserting a Node at the End of Linked List

```
Step 1 : IF AVAIL = NULL
         Write OVERFLOW
         Goto to step 10
     [ END OF IF ]
Step 2 : SET NEW_NODE = AVAIL
step 3 : SET AVAIL = AVAIL → NEXT
step 4 : SET NEW_NODE → DATA = VAL
Step 5 : SET NEW-NODE → NEXT = NULL
step 6 : SET PTR = START
step 7 : Repeat Step 8 while PTR → NEXT != NULL
step 8 : SET PTR = PTR → NEXT
         [ END OF LOOP ]
step 9 : SET PTR → NEXT = NEW_NODE
step 10 : EXIT
```

A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

Semester: 3 Subject: DSA Academic Year: 17-18

Inserting a Node After a Given Node in a Linked List

```
Step 1 : IF AVAIL = NULL
           write OVERFLOW
           GO to step 12
         [END OF IF]
Step 2 : SET NEW_NODE = AVAIL
Step 3 : SET AVAIL = AVAL → NEXT
Step 4 : SET NEW_NODE → DATA = VAL
Step 5 : SET PTR = START
Step 6 : SET PREPTR = PTR
Step 7 : Repeat Step 8 and 9 while PREPTR→
           Data
           != NUM
Step 8 : SET PREPTR = PTR
Step 9 : SET PTR = PTR→ NEXT
         [END OF LOOP]
Step 10 : PREPTR → NEXT = NEW_NODE
Step 11 : SET NEW_NODE → NEXT = PTR
Step 12 : EXIT
```

Inserting a Node Before a Given Node in Linked List

```
Step 1 : IF AVAIL = NULL
           Write OVERFLOW
           Goto Step 12
         [END OF IF]
Step 2 : SET NEW_NODE = AVAIL
Step 3 : SET AVAIL = AVAIL → NEXT
Step 4 : SET NEW_NODE → DATA = VAL
Step 5 : SET PTR = START
Step 6 : SET PREPTR = PTR
Step 7 : Repeat step 8 and 9 while
           PTR → DATA != NUM
Step 8 : SET PREPTR = PTR
Step 9 : SET PTR = PTR → NEXT
         [END OF LOOP]
Step 10 : PREPTR → NEXT = NEW_NODE
Step 11 : SET NEW_NODE → NEXT = PTR
Step 12 : EXIT
```

[Algorithm to insert a new node before a node that has NUM.]

A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

Semester: 3 Subject: DSA Academic Year: 17-18

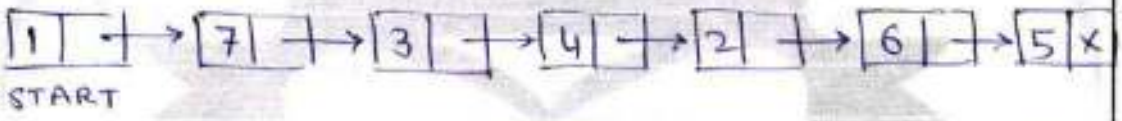
Deleting a Node from a Linked List

In this section, we will discuss how a node is deleted from an already existing linked list. We will consider three cases and then see how deletion is done in each case

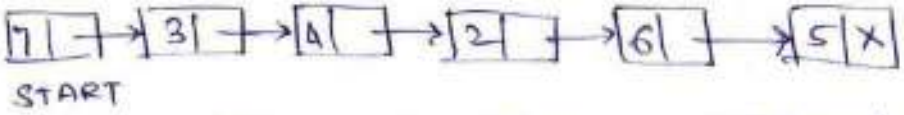
Case 1 : The first node is deleted
Case 2 : The last node is deleted
Case 3 : The node after a given node is deleted

Deleting the first Node from a Linked List

When we want to delete a node from the beginning of the list, then the following changes will be done in the linked list.



Make START to point to the next node in sequence



Deleting the first node of a linked list

```
step 1 : IF START = NULL
         Write UNDERFLOW
         Goto step 5
         [END OF IF]
step 2 : SET PTR = START
step 3 : SET START = START → NEXT
step 4 : FREE PTR
step 5 : EXIT
```

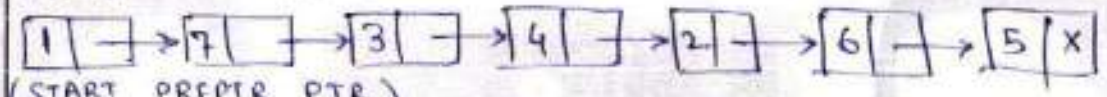
Vidyavardhini's Bharatiya Vidya Mandir's
A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

Semester: 3 Subject: DSA Academic Year: 17-18

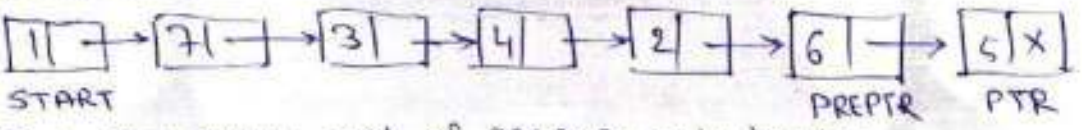
Deleting the last Node from a Linked List

Suppose we want to delete the last node from Linked list, then the following changes will be done in the linked list.

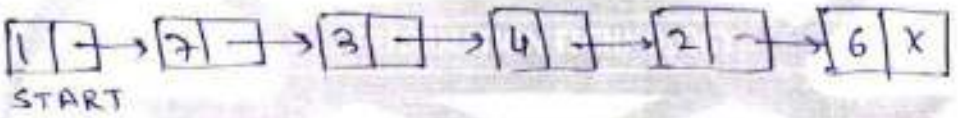
Take a pointer variables PTR & PREPTR which initially point to START



Move PTR and PREPTR such that NEXT Part of PTR = NULL. PREPTR always points to the node just before the node pointed by PTR



Set the NEXT part of PREPTR node to NULL.



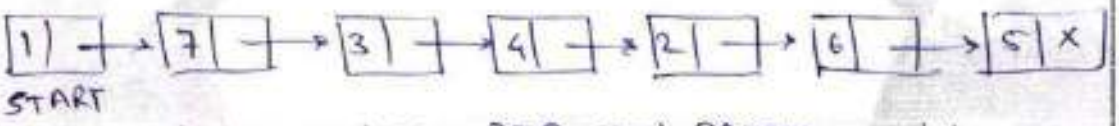

```
Step 1 : IF START = NULL
         Write UNDERFLOW
         GOTO step 8
         [END OF IF]
Step 2 : SET PTR = START
Step 3 : Repeat Step 4 & 5 while PTR→NEXT != NULL
Step 4 :   SET PREPTR = PTR
Step 5 :   SET PTR = PTR → NEXT
         [END OF LOOP]
Step 6 : SET PREPTR → NEXT = NULL
Step 7 : FREE PTR
Step 8 : EXIT
```

A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

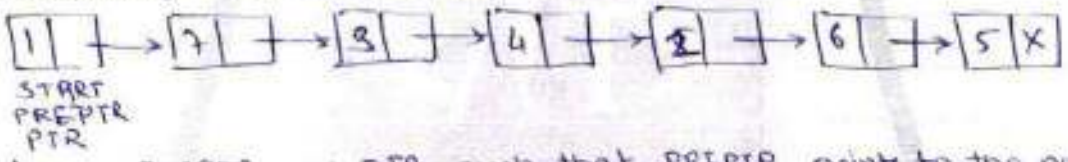
Semester: 3 Subject: PSA Academic Year: 17-18

Deleting the Node After a given Node in a Linked List

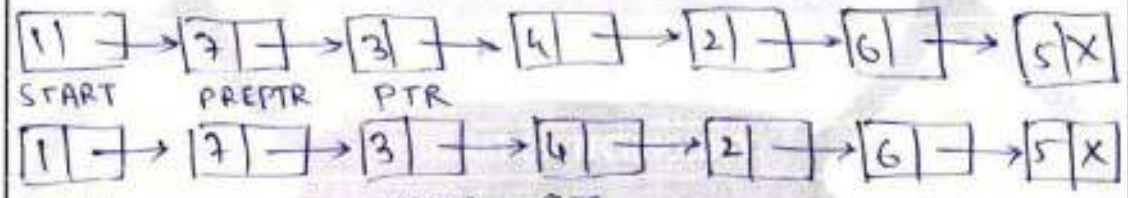
Suppose we want to delete the node that succeeds the node which contains data value 4 then the following changes will be done in the linked list.



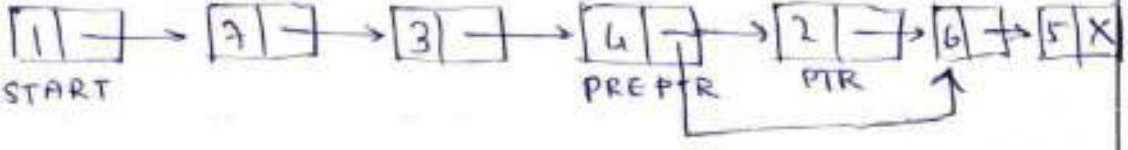
Take pointer variable PTR and PREPTR which initially point to start.

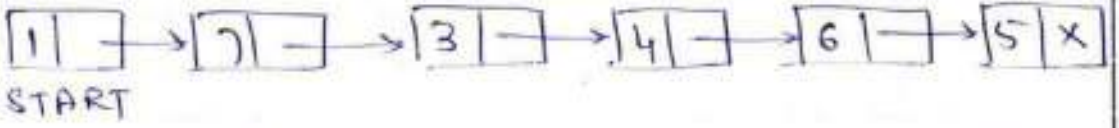


Move PREPTR and PTR such that PREPTR points to the node containing VAL and PTR points to the succeeding



SET THE NEXT part of PREPTR to the NEXT part of PTR





Deleting the node after a given node in linked list

Step 1 : IF START = NULL
WRITE UNDERFLOW
Go to Step 10
[END OF IF]
Step 2 : SET PTR = START
Step 3 : SET PREPTR = PTR
Step 4 : Repeat Step 5 and 6 while
PREPTR -> DATA != NUM
step 5 : SET PREPTR = PTR
step 6 : SET PTR = PTR -> NEXT
[END OF LOOP]
step 7 : SET TEMP = PTR
step 8 : SET PREPTR -> NEXT = PTR -> NEXT
step 9 : FREE TEMP
Step 10 : EXIT

Alogorithm to delete the node after a given node

//Creating a Singly Linked List inserting a node at the beginning,At end and at specific position.Also deleting a node from begining,from end and from specific position.

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

// A linked list (LL) node to store a data element
struct node
{
 int data;
 struct node* next;
}*start;

//Insert data at the beginning
 void insert_at_beg(int x)
 {
  /*Allocate Node or allocate memory for new node*/
  struct node* newnode=(struct node*)malloc(sizeof(struct node));
  printf("\nEnter the data :");
  scanf("%d",&x);
  if(start==NULL)//if Linked List is empty
  {
   /*Put in the data in the node and set next to NULL*/
    newnode->data=x;
    newnode->next=NULL;
   /*Move start to point to the newnode*/
    start=newnode;
  }
  else
  {
  /*Put in the data in the node*/
   newnode->data=x;
  /*Make next of new node as start*/
   newnode->next=start;
  /*Move the start to point to the new node */
   start=newnode;
  }
 }

//Insert data at the end
 void insert_at_end(int x)
 {
  /*create a pointer to traverse till end of linked list*/
  struct node* ptr;

   /*Allocate Node or allocate memory for new node*/
  struct node* newnode=(struct node*)malloc(sizeof(struct node));
  printf("\nEnter the data :");
  scanf("%d",&x);

  /*Put in the data in the node and set next to NULL*/
  newnode->data=x;
  newnode->next=NULL;

  if(start==NULL)//if Linked List is empty Make next of new node as start
```

```
 {  //newnode->data=x;
    //newnode->next=NULL;

    /*Move the start to point to the new node*/
    start=newnode;
 }
 else //if not empty then
 {
  ptr=start;
  while(ptr->next!=NULL)//traverse through Linked List till last node
    ptr=ptr->next;

  ptr->next=newnode;//linked last node of next to newnode
 }
}

//Insert data at specific position
 void insert_at_specific_position(int x,int pos)
 {
  struct node* ptr;
  ptr=start;
  int i=0;
  struct node*newnode=(struct node*)malloc(sizeof(struct node));
  printf("\nEnter the data :");
  scanf("%d",&x);
  printf("Enter the position :");
  scanf("%d",&pos);
  if(pos==1)
  {
    printf("Please use insert_at_begin() ");
    //break;
   }
   else //if posotion is not 1 then
   {
       newnode->data=x;
       if(start==NULL)//if LL is empty
       {
         start=newnode;
         newnode->next=NULL;
       }
       else
       {
          while(i<(pos-2))//if pos=5 while loop will continue from 0 till i<3
          {
           ptr=ptr->next;//in last iteration ptr=3->next i.e 4
           i++;
          }
          newnode->next=ptr->next;//ptr->next is 4->next i.e 5 that is 5th node shifted to        6th position
          ptr->next=newnode;//4->next is set to newnode i.e newnode is added at 5th position
       }
  }
 }
//Delete data from beginning
 void delete_from_beg()
 {
```

```
 struct node* ptr;
 if(start==NULL)
 {
  printf("\nUNDERFLOW ! ! !");
  //break;
 }
 else
 {
  ptr=start;
  start=start->next;
  printf("\n%d is deleted...",ptr->data);
  free(ptr);
 }
}

//Delete data from end
 void delete_from_end()
 {
  struct node* ptr,*preptr;
  if(start==NULL)
  {
   printf("\nUNDERFLOW ! ! !");
   //break;
  }
  else
  {
   ptr=start;
   if(ptr->next==NULL)//if there is only one element in the LL
   {
    printf("\n%d is deleted...",ptr->data);
    start=NULL;
    free(ptr);
   }
   else
   {
    while(ptr->next!=NULL)
    {
     preptr=ptr;//preptr=1
     ptr=ptr->next;//ptr=ptr->next=1->next=2
    }
    preptr->next=NULL;
    printf("\n%d is deleted...",ptr->data);
    free(ptr);
   }
  }
 }

//Delete data from specific position
 void delete_from_specific_position(int pos)
 {
  struct node* ptr,* temp;
  int i=0;
  if(start==NULL)
  {
   printf("\nUNDERFLOW ! ! !");
```

```
 //break;
 }
 else
 {
 printf("\nEnter the position :");
 scanf("%d",&pos);
 ptr=start;
 if(ptr->next==NULL)//if there is only one node in the LL
 {
 printf("\n%d is deleted...",ptr->data);
 start=NULL;
 free(ptr);
 }
 else
 {
 while(i<(pos-1))//if pos=4 then while loop will continue from 0 to 3
 {
 temp=ptr;//in last iteration temp=4
 ptr=ptr->next;//ptr=4->next=5
 i++;
 }
 temp->next=ptr->next;//4->next=5->next i.e. 4->next=6
 printf("\n%d is deleted...",ptr->data);
 free(ptr);
 }
 }
}
void display()
{
 struct node *ptr=start;
 if(ptr==NULL)
 {
 printf("\nSORRY ! NO ELEMENT ! ! !");
 }
 else
 {
 while(ptr!=NULL)
 {
 printf("%d -> ",ptr->data);
 ptr=ptr->next;
 }
 }
}
void main()
{
 start=NULL;
 //head;
 int ch,x,pos;
 int z=1;

while(z)  //While loop to keep program in loop
{
 printf("\n\n----------------------------------------\n");

 printf("1. Insert at the begining\n");
```

```
printf("2. Insert at the end\n");

printf("3. Insert at a specific position\n");

printf("4. Delete from begining\n");

printf("5. Delete from the end\n");

printf("6. Delete from a specific position\n");

printf("7. Display\n");

printf("8. Exit");

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

printf("ENTER A CHOICE :");
scanf("%d",&ch);
switch(ch)
{
 case 1: insert_at_beg(x);
         break;
 case 2: insert_at_end(x);
         break;
 case 3: insert_at_specific_position(x,pos);
         break;
 case 4: delete_from_beg();
         break;
 case 5: delete_from_end();
         break;
 case 6: delete_from_specific_position(pos);
         break;
 case 7: display();
         break;
 case 8: exit(0);
         break;
 default : printf("\nOOPS ! WRONG CHOICE !");
          break;
 }
 //goto head;
printf(" \n Do you want to cotin....? press 1 or 0 ");
scanf("%d",&x);
}//end of while loop
}//end of main()
```

//Perform Operations on Singly Linked List like copy,concat,count no of nodes, split and reverse.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
 int data;
 struct node *next;
}*start;

void copy()
{
 struct node *ptr=start,*t=NULL;
 if(start->next==NULL)//only one element in LL
 {
  struct node* temp=(struct node*)malloc(sizeof(struct node));
  temp->data=start->data;
  temp->next=NULL;
  t=temp;
 }
 else//if LL has more elements
 {
  while(ptr!=NULL)
  {
   struct node* temp=(struct node*)malloc(sizeof(struct node));
   temp->data=ptr->data;
   temp->next=NULL;
   if(t==NULL)//assign t to start of copied LL
   {
    t=temp;//if copied LL is empty then assign t to temp
   }
   else// if t is not empty then traverse till the end
   {
    struct node *add=t;//pointer to link list "t"
    while(add->next!=NULL)
    {
     add=add->next;
    }
    add->next=temp;
   }
   ptr=ptr->next;//to copy next node from original LL to copied LL
  }
 }
 printf("\n->Copied list:\n");
 struct node *k=t;//pointer to link list "t"
 while(k!=NULL)//display copied LL using while loop
   {
    printf("%d ",k->data);
    k=k->next;
   }
}
void count()
{
 struct node *ptr=start;
 int k=0;
```

```
 while(ptr->next!=NULL)
 {
  k++;
  ptr=ptr->next;
 }
 k++;//as intially k=0 so incermented by 1 to count the last node
 printf("\n->Length of the list : %d",k);
}
void concatenate()
{
 struct node *head=NULL;
 int m,j,x;
 printf("Create list2:-\n");
 printf("\nEnter the no. of nodes : ");
 scanf("%d",&m);

 for(j=0;j<m;j++)
 {
  struct node* temp=(struct node*)malloc(sizeof(struct node));
  printf("\nEnter the data %d : ",j+1);
  scanf("%d",&x);
  temp->data=x;
  temp->next=NULL;
  if(head==NULL)//for first node of empty LL
  {
   head=temp;
  }
  else //for next nodes
  {
   struct node *add=head;
   while(add->next!=NULL)//traverse till the last node
   {
     add=add->next;
   }
   add->next=temp;
  }
 }//linked list2 created
 struct node *ptr1=start,*ptr2=head;
 while(ptr1->next!=NULL)//traverse till last node of LL1
 {
   ptr1=ptr1->next;
 }

 ptr1->next=ptr2;//set last node ptr of LL1 to first node of LL2

 printf("\n->Concatenated List :-\n");
 struct node *k=start;
 while(k!=NULL)
   {
    printf("%d ",k->data);
    k=k->next;
   }
}
void split()
{
```

```
struct node *st,*ptr1,*ptr=start,*head=NULL,*k=start;
int p=1,pos;
printf("\nEnter the position of split : ");
scanf("%d",&pos);
while(ptr->next!=NULL)
{
 ptr=ptr->next;
 p++;
 if(p==pos)//if pos=3 when we reach to third node we store its next ptr in "st"
            // and set 3->next to NULL
 {
  st=ptr->next;
  ptr->next=NULL;
 }
}
while(st!=NULL)
{
 if(head==NULL)//assign head as starting point of LL2
 {
  struct node* temp=(struct node*)malloc(sizeof(struct node));
  temp->data=st->data;
  temp->next=NULL;
  head=temp;
  st=st->next;
 }
 else //if head is not null then
 {
  struct node* temp=(struct node*)malloc(sizeof(struct node));
  temp->data=st->data;
  temp->next=NULL;
  ptr1=head;
  while(ptr1->next!=NULL)//travese till the end of new LL
  {
   ptr1=ptr1->next;
  }
  ptr1->next=temp;
  st=st->next;
 }
}
printf("\nList 1:-\n");
while(k!=NULL)
{
 printf("%d ",k->data);
 k=k->next;
}
struct node *j=head;
printf("\nList 2:-\n");
while(j!=NULL)
{
 printf("%d ",j->data);
 j=j->next;
}
}
void reverse()
{
```

```
 struct node *head=NULL,*ptr=start;
 struct node* temp=(struct node*)malloc(sizeof(struct node));
 while(ptr!=NULL)
 {
  if(head==NULL)//if LL is empty
  {
   temp->data=ptr->data;
   temp->next=NULL;
   head=temp;
  }
  else //if LL has more nodes
  {
   struct node* temp=(struct node*)malloc(sizeof(struct node));
   temp->data=ptr->data;
   temp->next=head;
   head=temp;
  }
  ptr=ptr->next;
 }
 struct node *k=head;
 while(k!=NULL)
 {
  printf("%d ",k->data);
  k=k->next;
 }
}
void main()
{
 start=NULL;
 int ch,n,x,i;
 printf("Create list1:-\n");
 printf("\nEnter the no. of nodes : ");
 scanf("%d",&n);

 for(i=0;i<n;i++)
 {
  struct node* newnode=(struct node*)malloc(sizeof(struct node));
  printf("\nEnter the data %d : ",i+1);
  scanf("%d",&x);
  newnode->data=x;
  newnode->next=NULL;
  if(start==NULL)//if the LL is empty
  {
   start=newnode;
  }
  else
  {
   struct node *add=start;

   while(add->next!=NULL)//traverse till the end of LL
   {
     add=add->next;
   }
   add->next=newnode;
  }
```

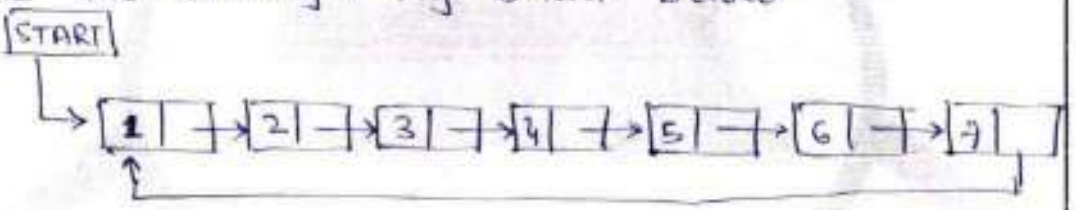
```
}
printf("\nLIST CREATED SUCCESFULLY !\n");

struct node *k=start;
while(k!=NULL)//display elements from LL
{
 printf("%d ",k->data);
 k=k->next;
}
do
{
printf("\n\n--------LINKED LIST OPERATION MENU--------\n");
printf("1. COPY\n2. CONCATENATE\n3. SPLIT\n4. COUNT\n5. REVERSE\n6. EXIT");
printf("\n-----------------------------------\n");
printf("ENTER YOUR CHOICE : ");
scanf("%d",&ch);
switch (ch)
{
 case 1: copy();
         break;
 case 2: concatenate();
         break;
 case 3: split();
         break;
 case 4: count();
         break;
 case 5: reverse();
         break;
 case 6: exit(0);
         break;
 default : printf("OOPS ! WRONG CHOICE !");
          break;
}
 }while(ch!=6);
}
```

A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

Semester: 3 Subject: DSA Academic Year: 17-18

Circular Linked Lists

In a circular linked list, the last node contains a pointer to the first node of the list. We can have a circular singly linked list as well as a circular doubly linked list. While traversing a circular linked list, we can begin at any node and traverse the list in any direction forward or backward until we reach the the same node where we started. Thus a circular linked list has no beginning and no ending. Fig shown below



Inserting a New Node in a Circular Linked list

In this section, we will see how a new node is added into an already existing linked list. We will take two cases and then see how insertion is done in each case.

Case 1 : The new node is inserted at the beginning of the circular linked list

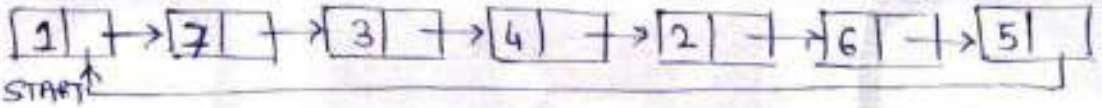
Case 2 : The new node is inserted at the end of the circular linked list.

Parshvanath Charitable Trust's
A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

Semester: 3 Subject: DSA Academic Year: 17-18

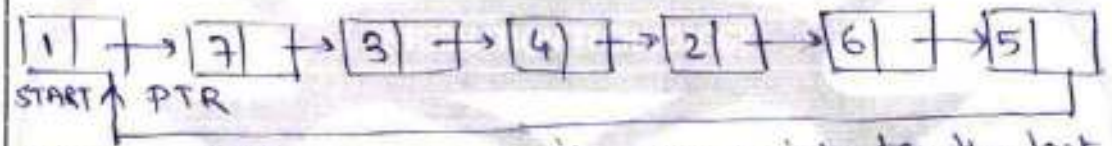
Inserting a Node at the Beginning of a Circular Linked List.

Consider the linked list shown in following fig. Suppose we want to add a new node with data 9 as the first node of list. Then the following changing will be in the linked list.

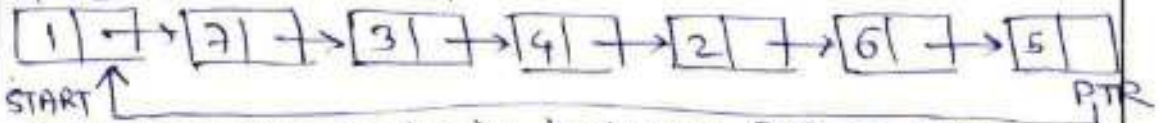


Allocate memory for the new node and initialize its DATA part to 9

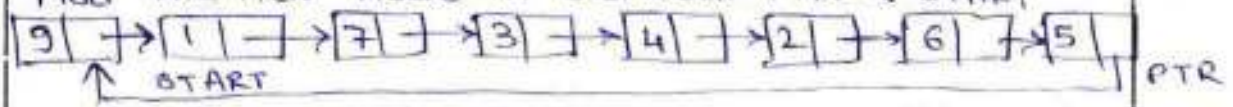
Take a pointer variable PTR that points to the START node of the list.



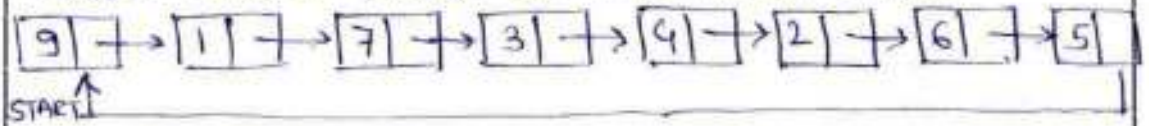
Move PTR so that it now points to the last node of the list.



Add the new node in between PTR & START



Make START point to the new node



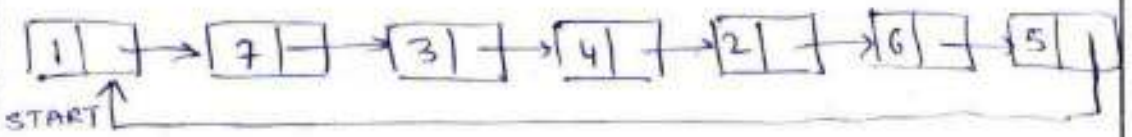
Inserting a new node at the beginning of circular linked list.

```
Step 1: IF AVAIL = NULL
        Write OVERFLOW
        Go to Step 11
    [END OF IF]
Step 2: SET NEW_NODE = AVAIL
Step 3: SET AVAIL = AVAIL → NEXT
Step 4: SET NEW_NODE → DATA = VAL
Step 5: SET PTR = START
Step 6: Repeat Step 7 while PTR →
        NEXT != START
Step 7: PTR = PTR → NEXT
    [END OF LOOP]
Step 8: SET NEW_NODE → NEXT =
        START
Step 9: SET PTR → NEXT = NEW_NODE
Step 10: SET START = NEW_NODE
Step 11: EXIT
```

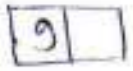
Algorithm to insert a new node at the beginning

Inserting a Node at the END of a circular Linked List

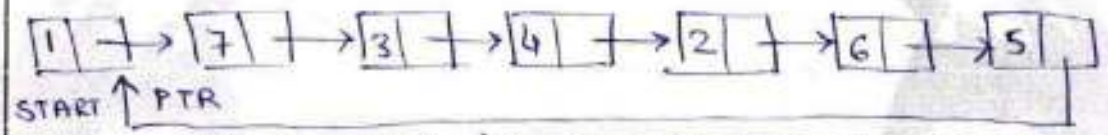
Suppose we want to add a new node with data 9 as the last node of the list. Then the following changes will be done in the linked list



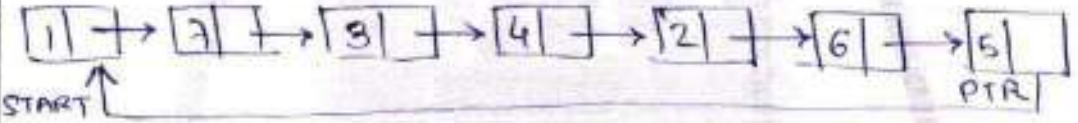
Allocate memory for the new node and initialize its DATA part to 9.



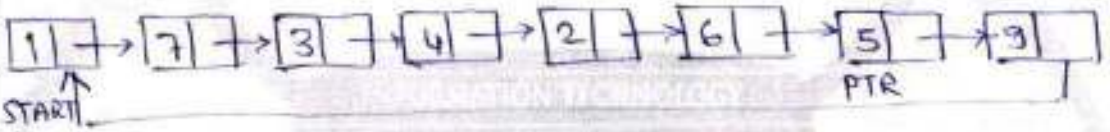
Take a pointer variable PTR which will initially point to START



Move PTR so that it now points to the least node of the list.



Add the new node after the node pointed by PTR.




```
Step 1 : IF AVAIL = NULL
         Write OVERFLOW
         Go to Step 10
         [END OF IF]
Step 2 : SET NEW_NODE = AVAIL
Step 3 : SET AVAIL = AVAIL → NEXT
Step 4 : SET NEW_NODE → DATA = VAL
STEP 5 : SET NEW_NODE → NEXT = START
STEP 6 : SET PTR = START
Step 7 : Repeat Step 8 while PTR → NEXT !=
Step 8 : SET PTR = PTR → NEXT
         [END OF LOOP]
Step 9 : SET PTR → NEXT = NEW_NODE
Step 10 : EXIT
```

A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

Semester: 3 Subject: DSA Academic Year: 17-18

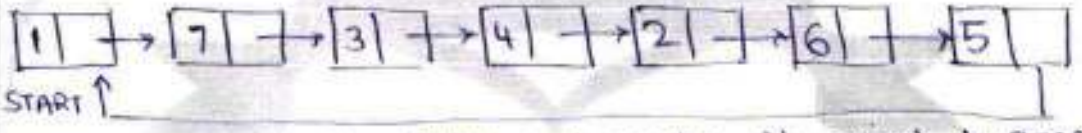
Deleting a Node from a Circular Linked List

In this section, we will discuss how a node is deleted from an already existing circular linked list. We will take two cases and then see how deletion is done in each case. Rest of the cases of deletion are same as that given for singly linked list.

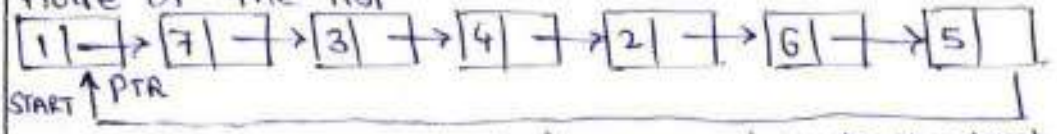
Case 1 : The first node is deleted
Case 2 : The last node is deleted

Deleting the First Node from a Circular Linked List

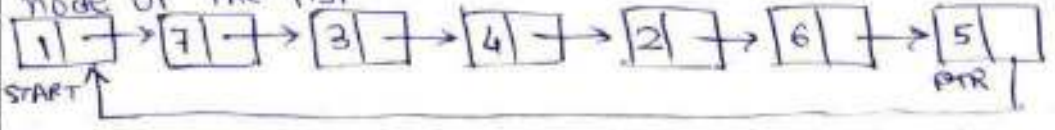
Consider the circular linked list shown in following fig. When we want to delete a node from the beginning of the list, then the following changes will be done in the linked list



Take a variable PTR and make it point to START node of the list

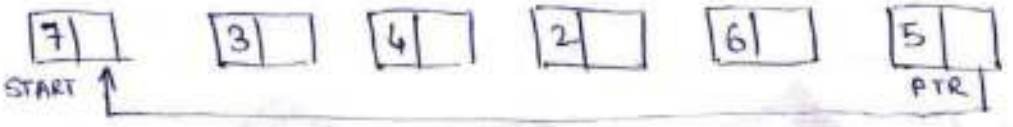


Move PTR further so that it now points to the last node of the list



The NEXT part of PTR is made to point to the second node of the list and the memory of

the first node is freed. The second node becomes the first node of the list.

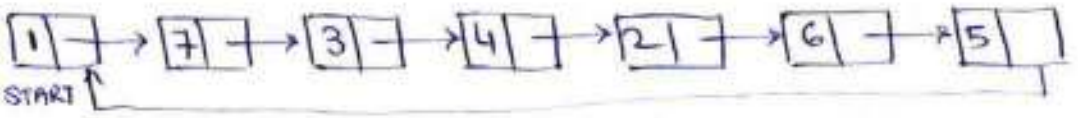



```
Step 1 : IF START = NULL
         Write UNDERFLOW
         Go to Step 8
       [END OF IF]
Step 2 : SET PTR = START
Step 3 : Repeat Step 4 while PTR → NEXT
           != START
Step 4 : SET PTR = PTR → NEXT
       [END OF LOOP]
Step 5 : SET PTR → NEXT = START → NEXT
Step 6 : FREE START
Step 7 : SET START = PTR → NEXT
Step 8 : EXIT
```

Algorithm to delete the first node

Deleting the Last Node from a Circular Linked List

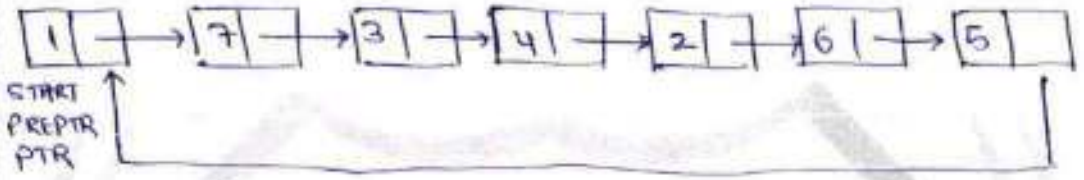
Consider the circular linked list shown in following fig. Suppose we want to delete the last node from the linked list, then the folbowing changes will be done in the linked list.



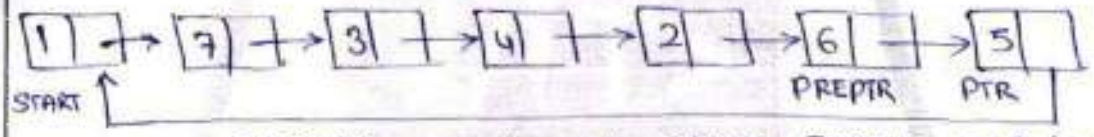
A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

Semester: 3 Subject: DSA Academic Year: 17-18

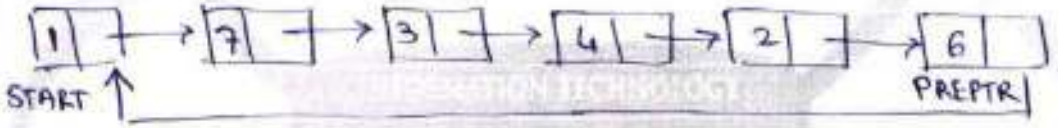
Take two pointer PREPTR and PTR which will initially point to START



Move PTR so that it points to the last node of the list. PREPTR will always point to the node preceding PTR.



Make the PREPTR's next part store START node's address and free the space allocated for PTR Now PREPTR is last node of the list




```
step 1 : IF START = NULL
           Write UNDERFLOW
           Go to Step 8
         [END OF IF]
Step 2 : SET PTR = START
step 3 : Repeat Step 4 and 5 while
           PTR → NEXT != START
step 4 : SET PREPTR = PTR
step 5 : SET PTR = PTR → NEXT
         [END OF LOOP]
Step 6 : SET PREPTR → NEXT = START
step 7 : FREE PTR
step 8 : EXIT
```

```
/*
1.  * C Program to Demonstrate Circular Single Linked List
2.  */



3. #include <stdio.h>
4. #include <stdlib.h>
5.
6. struct node
7. {
8.     int data;
9.     struct node *link;
10. };
11.
12. struct node *head = NULL, *x, *y, *z;
13.
14. void create();
15. void ins_at_beg();
16. void ins_at_pos();
17. void del_at_beg();
18. void del_at_pos();
19. void traverse();
20. void search();
21. void sort();
22. void update();
23. void rev_traverse(struct node *p);
24.
25. void main()
26. {
27.     int ch;
28.
29.     printf("\n 1.Creation \n 2.Insertion at beginning \n 3.Insertion at
remaining");
30.     printf("\n4.Deletion at beginning \n5.Deletion at remaining
\n6.traverse");
31.     printf("\n7.Search\n8.sort\n9.update\n10.Exit\n");
32.     while (1)
33.     {
34.         printf("\n Enter your choice:");
35.         scanf("%d", &ch);
36.         switch(ch)
37.         {
38.         case 1:
39.             create();
40.             break;
41.         case 2:
42.             ins_at_beg();
43.             break;
44.         case 3:
45.             ins_at_pos();
46.             break;
47.         case 4:
48.             del_at_beg();
49.             break;
50.         case 5:
```

```
            del_at_pos();
            break;
        case 6:
            traverse();
            break;
        case 7:
            search();
            break;
        case 8:
            sort();
            break;
        case 9:
            update();
            break;
        case 10:
            rev_traverse(head);
            break;
        default:
            exit(0);
        }
    }
}

/*Function to create a new circular linked list*/
void create()
{
    int c;

    x = (struct node*)malloc(sizeof(struct node));
    printf("\n Enter the data:");
    scanf("%d", &x->data);
    x->link = x;
    head = x;
    printf("\n If you wish to continue press 1 otherwise 0:");
    scanf("%d", &c);
    while (c != 0)
    {
        y = (struct node*)malloc(sizeof(struct node));
        printf("\n Enter the data:");
        scanf("%d", &y->data);
        x->link = y;
        y->link = head;
        x = y;
        printf("\n If you wish to continue press 1 otherwise 0:");
        scanf("%d", &c);
    }
}

/*Function to insert an element at the begining of the list*/

void ins_at_beg()
{
     x = head;
     y = (struct node*)malloc(sizeof(struct node));
     printf("\n Enter the data:");
     scanf("%d", &y->data);
```

```
    while (x->link != head)
    {
        x = x->link;
    }
    x->link = y;
    y->link = head;
    head = y;
}

/*Function to insert an element at any position the list*/

void ins_at_pos()
{
    struct node *ptr;
    int c = 1, pos, count = 1;

    y = (struct node*)malloc(sizeof(struct node));
    if (head == NULL)
    {
        printf("cannot enter an element at this place");
    }
    printf("\n Enter the data:");
    scanf("%d", &y->data);
    printf("\n Enter the position to be inserted:");
    scanf("%d", &pos);
    x = head;
    ptr = head;
    while (ptr->link != head)
    {
        count++;
        ptr = ptr->link;
    }
    count++;
    if (pos > count)
    {
        printf("OUT OF BOUND");
        return;
    }
    while (c < pos)
    {
        z = x;
        x = x->link;
        c++;
    }
    y->link = x;
    z->link = y;
}

/*Function to delete an element at any begining of the list*/

void del_at_beg()
{
    if (head == NULL)
        printf("\n List is empty");
    else
    {
```