Experiment 4.1

Student Name: Pranjal Kumar
UID: 20BCS3504
Branch: CSE
Section/Group: 607 B
Semester: 5th
Date of Performance: 06/08/2022
Subject Name: DAA Lab
Subject Code: 20CSP-312

1. Aim/Overview of the practical:-

Code to Insert and Delete an element at the beginning and at end in Doubly and Circular Linked List

2. Task to be done/ Which logistics used:-

Operation done in Insert and Delete an element at the beginning and at end in Doubly and Circular Linked List

3. Algorithm/Flowchart (For programming based labs):

Doubly Linked List

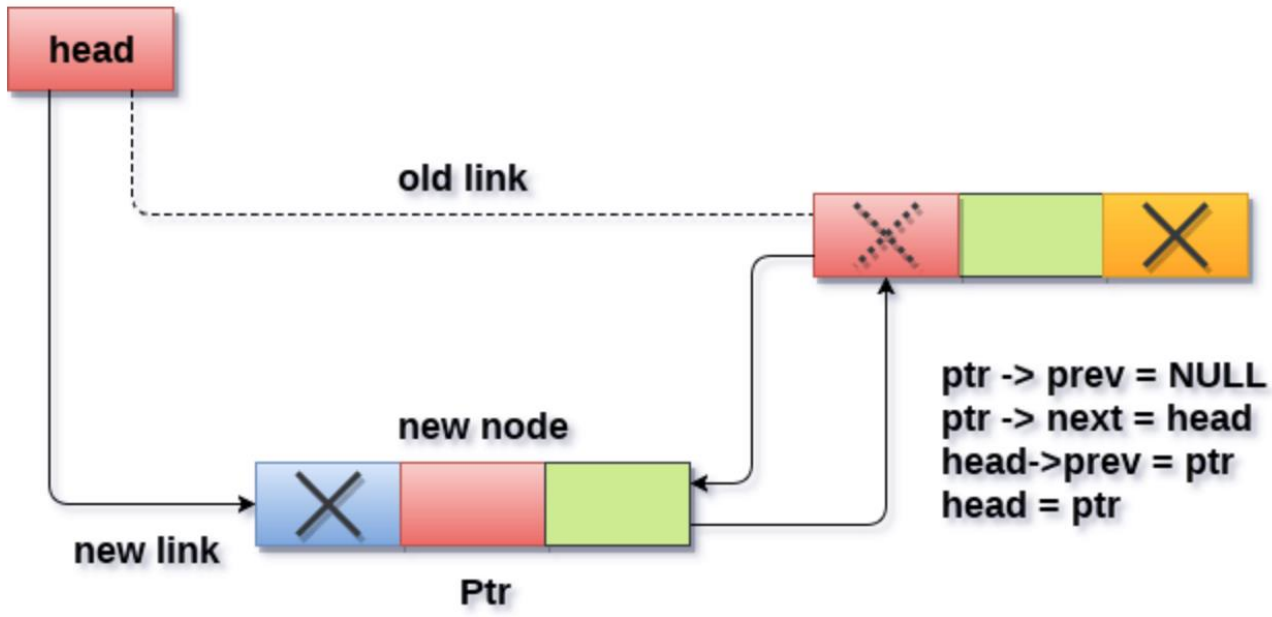
INSERTION:-
A) Insertion in doubly linked list at beginning

- **Step 1:** IF ptr = NULL

  Write OVERFLOW
  Go to Step 9
  [END OF IF]

- **Step 2:** SET NEW_NODE = ptr
- **Step 3:** SET ptr = ptr -> NEXT
- **Step 4:** SET NEW_NODE -> DATA = VAL
- **Step 5:** SET NEW_NODE -> PREV = NULL
- **Step 6:** SET NEW_NODE -> NEXT = START
- **Step 7:** SET head -> PREV = NEW_NODE
- **Step 8:** SET head = NEW_NODE

- **Step 9:** EXIT



B) Insertion in doubly linked list at the end

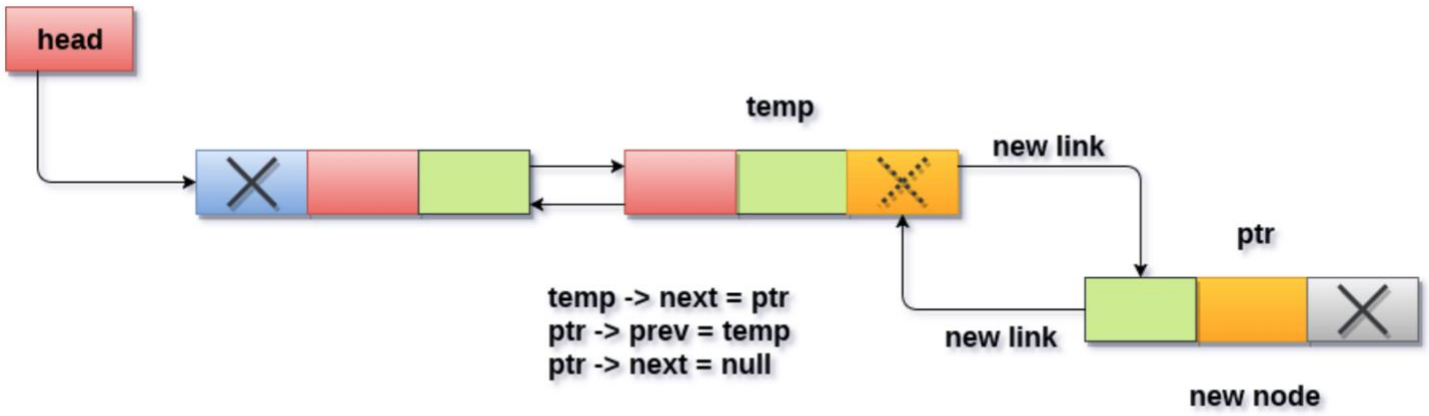
- **Step 1:** IF PTR = NULL

  Write OVERFLOW
  Go to Step 11
  [END OF IF]

- **Step 2:** SET NEW_NODE = PTR
- **Step 3:** SET PTR = PTR -> NEXT
- **Step 4:** SET NEW_NODE -> DATA = VAL
- **Step 5:** SET NEW_NODE -> NEXT = NULL
- **Step 6:** SET TEMP = START
- **Step 7:** Repeat Step 8 while TEMP -> NEXT != NULL
- **Step 8:** SET TEMP = TEMP -> NEXT

  [END OF LOOP]

- **Step 9:** SET TEMP -> NEXT = NEW_NODE
- **Step 10C:** SET NEW_NODE -> PREV = TEMP
- **Step 11:** EXIT



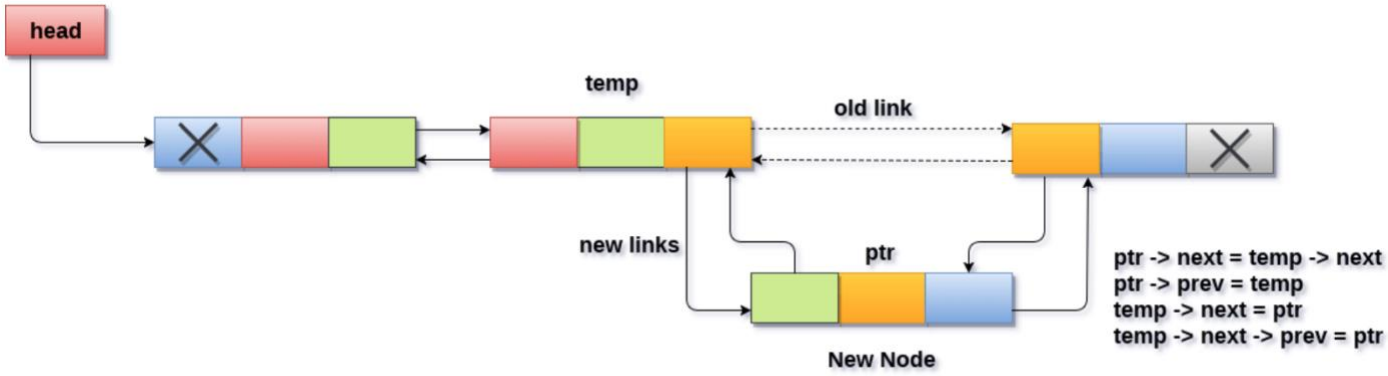
C) Insertion in doubly linked list after Specified node

- **Step 1:** IF PTR = NULL

  Write OVERFLOW
  Go to Step 15
  [END OF IF]
- **Step 2:** SET NEW_NODE = PTR
- **Step 3:** SET PTR = PTR -> NEXT
- **Step 4:** SET NEW_NODE -> DATA = VAL
- **Step 5:** SET TEMP = START
- **Step 6:** SET I = 0
- **Step 7:** REPEAT 8 to 10 until I<="" li="">
- **Step 8:** SET TEMP = TEMP -> NEXT
- **STEP 9:** IF TEMP = NULL
- **STEP 10:** WRITE "LESS THAN DESIRED NO. OF ELEMENTS"

GOTO STEP 15
[END OF IF]
[END OF LOOP]

- **Step 11:** SET NEW_NODE -> NEXT = TEMP -> NEXT
- **Step 12:** SET NEW_NODE -> PREV = TEMP
- **Step 13 :** SET TEMP -> NEXT = NEW_NODE
- **Step 14:** SET TEMP -> NEXT -> PREV = NEW_NODE
- **Step 15:** EXIT



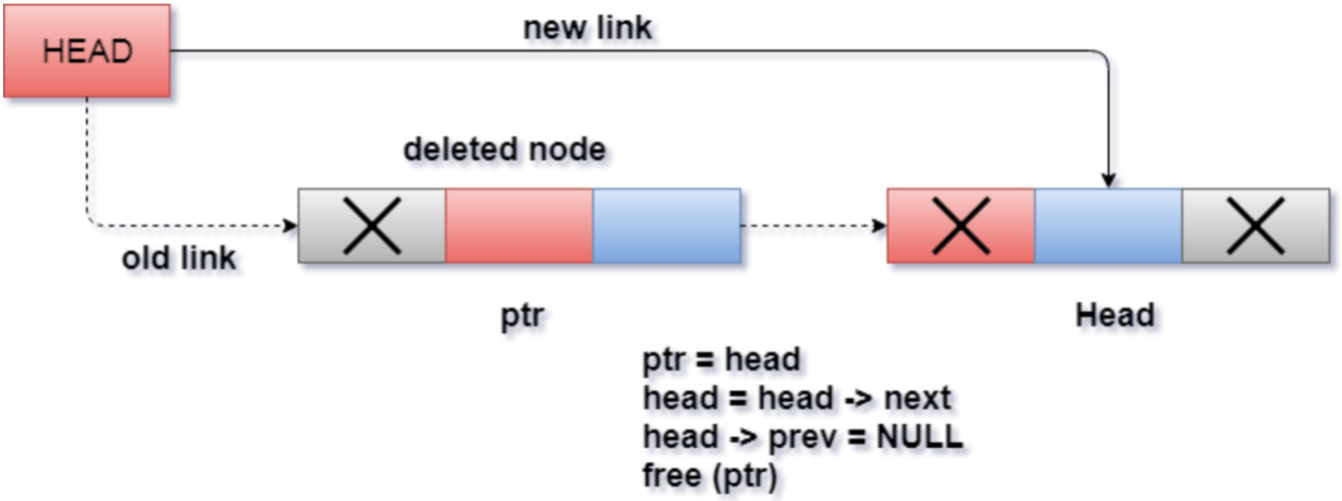
DELETION :

A) Deletion at beginning

- **STEP 1:** IF HEAD = NULL

WRITE UNDERFLOW
GOTO STEP 6

- **STEP 2:** SET PTR = HEAD
- **STEP 3:** SET HEAD = HEAD → NEXT
- **STEP 4:** SET HEAD → PREV = NULL
- **STEP 5:** FREE PTR
- **STEP 6:** EXIT



B) Deletion in doubly linked list at the end

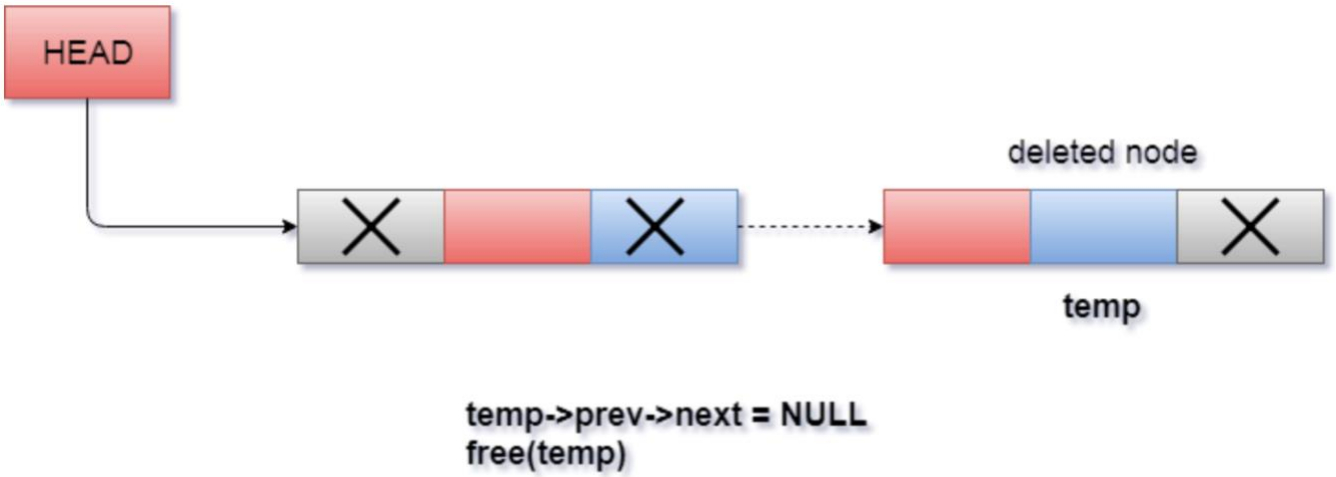
- **Step 1:** IF HEAD = NULL

  Write UNDERFLOW
  Go to Step 7
  [END OF IF]

- **Step 2:** SET TEMP = HEAD
- **Step 3:** REPEAT STEP 4 WHILE TEMP->NEXT != NULL
- **Step 4:** SET TEMP = TEMP->NEXT

  [END OF LOOP]

- **Step 5:** SET TEMP ->PREV-> NEXT = NULL
- **Step 6:** FREE TEMP
- **Step 7:** EXIT

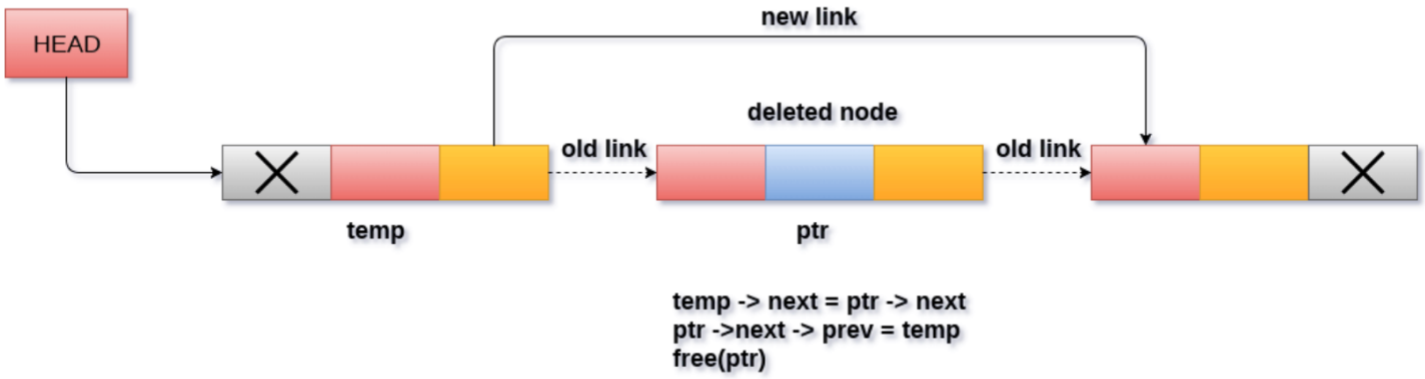


C) Deletion in doubly linked list after the specified node

- **Step 1:** IF HEAD = NULL

  Write UNDERFLOW
  Go to Step 9
  [END OF IF]
- **Step 2:** SET TEMP = HEAD
- **Step 3:** Repeat Step 4 while TEMP -> DATA != ITEM
- **Step 4:** SET TEMP = TEMP -> NEXT

  [END OF LOOP]
- **Step 5:** SET PTR = TEMP -> NEXT
- **Step 6:** SET TEMP -> NEXT = PTR -> NEXT
- **Step 7:** SET PTR -> NEXT -> PREV = TEMP
- **Step 8:** FREE PTR
- **Step 9:** EXIT



Circular Linked List

Insertion:-

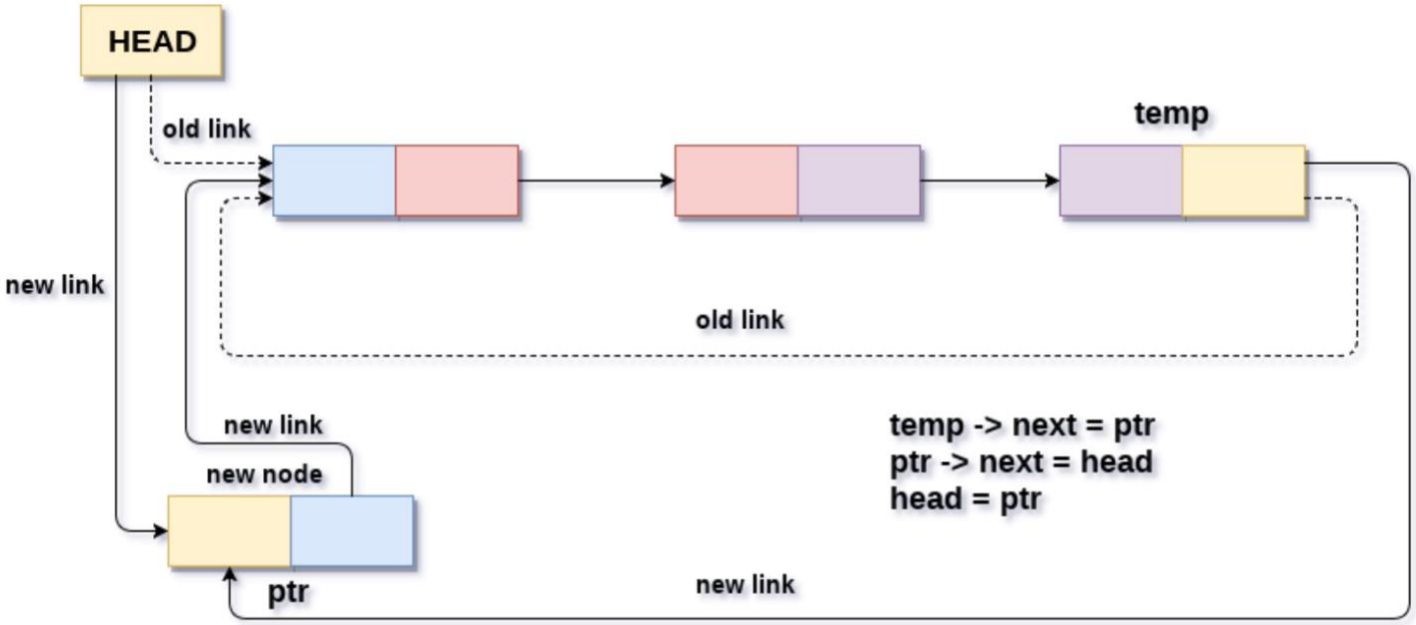
A) Insertion into circular singly linked list at beginning

- **Step 1:** IF PTR = NULL

  Write OVERFLOW
  Go to Step 11
  [END OF IF]
- **Step 2:** SET NEW_NODE = PTR
- **Step 3:** SET PTR = PTR -> NEXT
- **Step 4:** SET NEW_NODE -> DATA = VAL
- **Step 5:** SET TEMP = HEAD
- **Step 6:** Repeat Step 8 while TEMP -> NEXT != HEAD
- **Step 7:** SET TEMP = TEMP -> NEXT

  [END OF LOOP]
- **Step 8:** SET NEW_NODE -> NEXT = HEAD
- **Step 9:** SET TEMP → NEXT = NEW_NODE
- **Step 10:** SET HEAD = NEW_NODE

- **Step 11:** EXIT



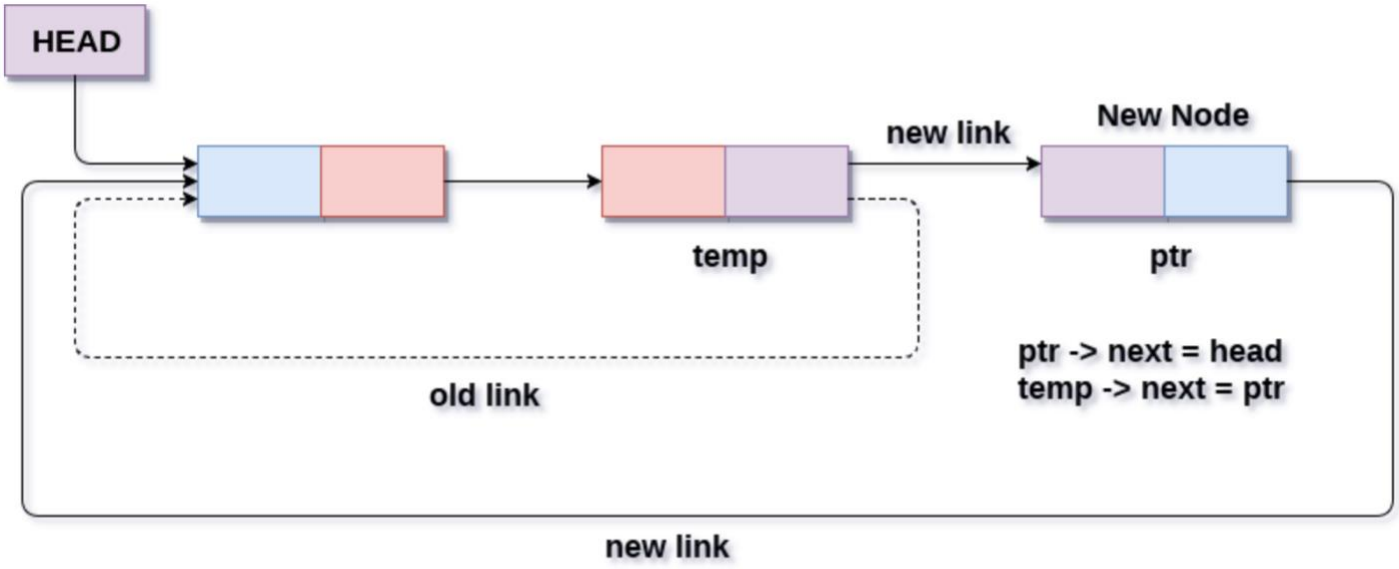
B) Insertion into circular singly linked list at the end

- **Step 1:** IF PTR = NULL

  Write OVERFLOW
  Go to Step 1
  [END OF IF]
- **Step 2:** SET NEW_NODE = PTR
- **Step 3:** SET PTR = PTR -> NEXT
- **Step 4:** SET NEW_NODE -> DATA = VAL
- **Step 5:** SET NEW_NODE -> NEXT = HEAD
- **Step 6:** SET TEMP = HEAD
- **Step 7:** Repeat Step 8 while TEMP -> NEXT != HEAD
- **Step 8:** SET TEMP = TEMP -> NEXT

  [END OF LOOP]

- **Step 9:** SET TEMP -> NEXT = NEW_NODE
- **Step 10:** EXIT



DELECTION:-

A) Deletion in circular singly linked list at beginning

- **Step 1:** IF HEAD = NULL

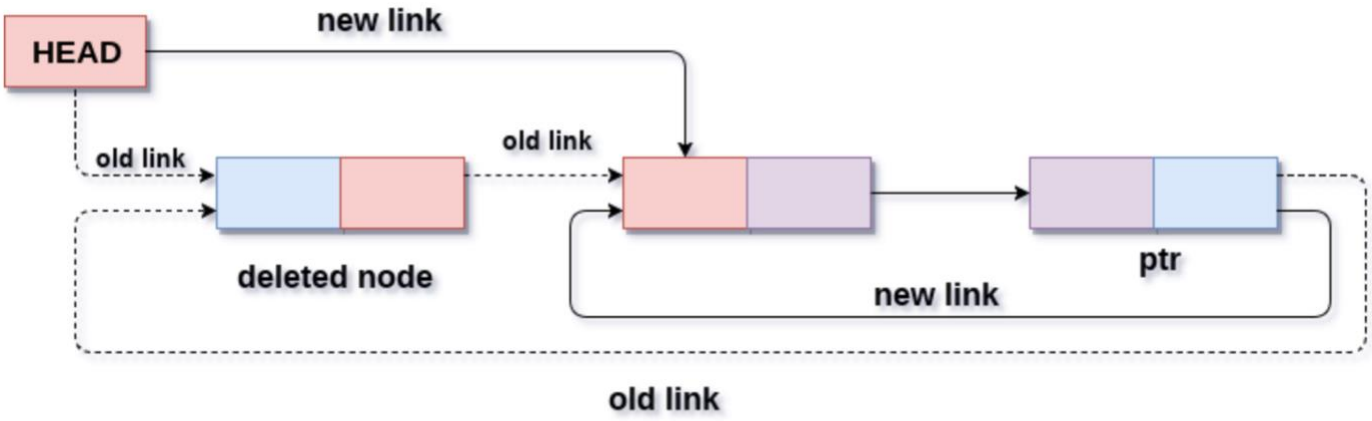
  Write UNDERFLOW
  Go to Step 8
  [END OF IF]

- **Step 2:** SET PTR = HEAD
- **Step 3:** Repeat Step 4 while PTR → NEXT != HEAD
- **Step 4:** SET PTR = PTR → next

  [END OF LOOP]

- **Step 5:** SET PTR → NEXT = HEAD → NEXT
- **Step 6:** FREE HEAD

- **Step 7:** SET HEAD = PTR → NEXT
- **Step 8:** EXIT




```
ptr -> next = head -> next
free head
head = ptr -> next
```

B) Deletion in Circular singly linked list at the end

- **Step 1:** IF HEAD = NULL

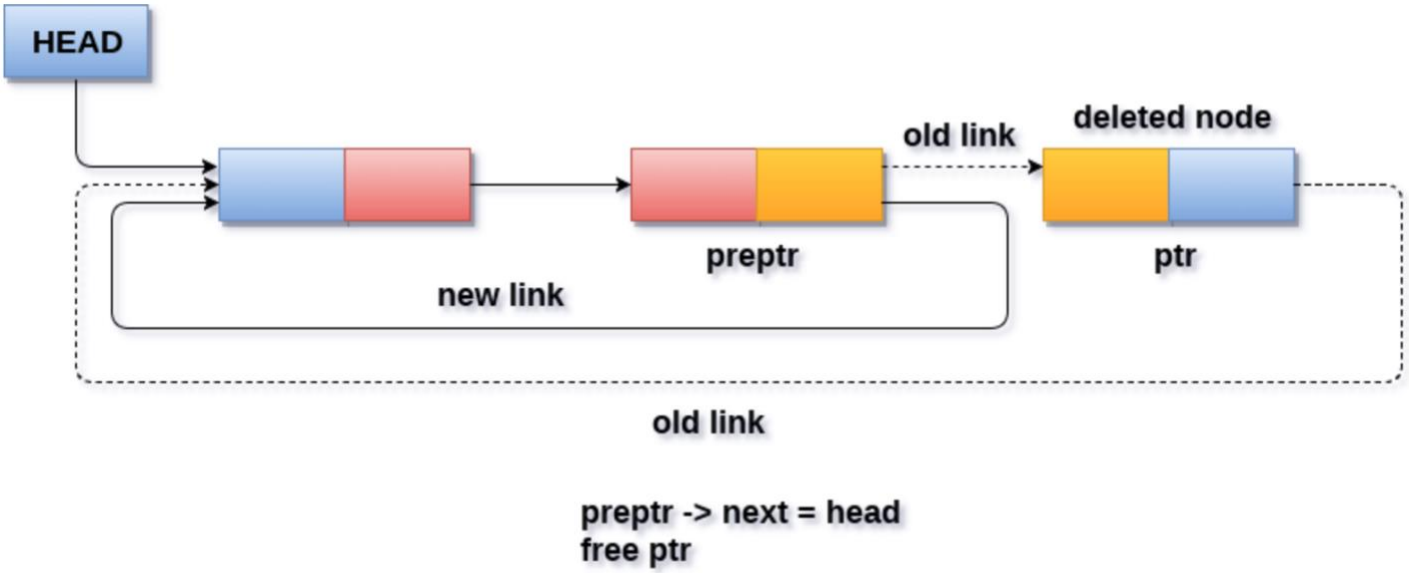
  Write UNDERFLOW
  Go to Step 8
  [END OF IF]

- **Step 2:** SET PTR = HEAD
- **Step 3:** Repeat Steps 4 and 5 while PTR -> NEXT != HEAD
- **Step 4:** SET PREPTR = PTR
- **Step 5:** SET PTR = PTR -> NEXT

  [END OF LOOP]

- **Step 6:** SET PREPTR -> NEXT = HEAD
- **Step 7:** FREE PTR

- **Step 8:** EXIT



4. Steps for experiment/practical/Code:-

Doubly Linked List

INSERTION:-

A) Insertion in doubly linked list at beginning

```
#include<stdio.h>
#include<stdlib.h>
void insertbeginning(int);
struct node
{
   int data;
   struct node *next;
   struct node *prev;
};
struct node *head;
void main ()
{
   int choice,item;
   do
```

```
    {
        printf("\nEnter the item which you want to insert?\n");
        scanf("%d",&item);
        insertbeginning(item);
        printf("\nPress 0 to insert more ?\n");
        scanf("%d",&choice);
    }while(choice == 0);
}
void insertbeginning(int item)
{

   struct node *ptr = (struct node *)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
      printf("\nOVERFLOW");
   }
   else
   {
   if(head==NULL)
   {
      ptr->next = NULL;
      ptr->prev=NULL;
      ptr->data=item;
      head=ptr;
   }
   else
    {
      ptr->data=item;
      ptr->prev=NULL;
      ptr->next = head;
      head->prev=ptr;
      head=ptr;
   }
 }
}
```

```c
#include<stdio.h>
#include<stdlib.h>
void insertbeginning(int);
struct node
{
    int data;
    struct node *next;
    struct node *prev;
};
struct node *head;
void main ()
{
    int choice,item;
    do
    {
        printf("\nEnter the item which you want to insert?\n");
        scanf("%d",&item);
        insertbeginning(item);
        printf("\nPress 0 to insert more ?\n");
        scanf("%d",&choice);
    }while(choice == 0);
}
void insertbeginning(int item)
{

   struct node *ptr = (struct node *)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
       printf("\nOVERFLOW");
   }
   else
   {
   if(head==NULL)
   {
       ptr->next = NULL;
       ptr->prev=NULL;
       ptr->data=item;
       head=ptr;
   }
   else
    {
       ptr->data=item;
       ptr->prev=NULL;
       ptr->next = head;
       head->prev=ptr;
       head=ptr;
   }
 }
}
```

B) Insertion in doubly linked list at the end

```
#include<stdio.h>
#include<stdlib.h>
void insertlast(int);
struct node
{
   int data;
   struct node *next;
   struct node *prev;
};
struct node *head;
void main ()
{
   int choice,item;
   do
   {
      printf("\nEnter the item which you want to insert?\n");
      scanf("%d",&item);
      insertlast(item);
      printf("\nPress 0 to insert more ?\n");
      scanf("%d",&choice);
   }while(choice == 0);
}
void insertlast(int item)
{

  struct node *ptr = (struct node *) malloc(sizeof(struct node));
  struct node *temp;
  if(ptr == NULL)
  {
     printf("\nOVERFLOW");

  }
  else
  {
      ptr->data=item;
     if(head == NULL)
     {
        ptr->next = NULL;
```

```
        ptr->prev = NULL;
        head = ptr;
      }
      else
      {
        temp = head;
        while(temp->next!=NULL)
        {
          temp = temp->next;
        }
        temp->next = ptr;
        ptr ->prev=temp;
        ptr->next = NULL;
      }
printf("\nNode Inserted\n");
   }
}
```

```c
#include<stdio.h>
#include<stdlib.h>
void insertlast(int);
struct node
{
    int data;
    struct node *next;
    struct node *prev;
};
struct node *head;
void main ()
{
    int choice,item;
    do
    {
        printf("\nEnter the item which you want to insert?\n");
        scanf("%d",&item);
        insertlast(item);
        printf("\nPress 0 to insert more ?\n");
        scanf("%d",&choice);
    }while(choice == 0);
}
void insertlast(int item)
{

   struct node *ptr = (struct node *) malloc(sizeof(struct node));
   struct node *temp;
   if(ptr == NULL)
   {
      printf("\nOVERFLOW");

   }
   else
```

```
    {
         ptr->data=item;
        if(head == NULL)
        {
            ptr->next = NULL;
            ptr->prev = NULL;
            head = ptr;
        }
        else
        {
            temp = head;
            while(temp->next!=NULL)
            {
                temp = temp->next;
            }
            temp->next = ptr;
            ptr ->prev=temp;
            ptr->next = NULL;
        }
printf("\nNode Inserted\n");
    }
}
```

C) Insertion in doubly linked list after Specified node

```
#include<stdio.h>
#include<stdlib.h>
void insert_specified(int);
void create(int);
struct node
{
   int data;
   struct node *next;
   struct node *prev;
};
struct node *head;
void main ()
{
   int choice,item,loc;
   do
   {
      printf("\nEnter the item which you want to insert?\n");
```

```
        scanf("%d",&item);
        if(head == NULL)
        {
           create(item);
        }
        else
        {
           insert_specified(item);
        }
        printf("\nPress 0 to insert more ?\n");
        scanf("%d",&choice);
    }while(choice == 0);
}
void create(int item)
    {
   struct node *ptr = (struct node *)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
      printf("\nOVERFLOW");
   }
   else
   {


   if(head==NULL)
   {
      ptr->next = NULL;
      ptr->prev=NULL;
      ptr->data=item;
      head=ptr;
   }
   else
   {
      ptr->data=item;printf("\nPress 0 to insert more ?\n");
      ptr->prev=NULL;
      ptr->next = head;
      head->prev=ptr;
      head=ptr;
   }
    printf("\nNode Inserted\n");
}

}
void insert_specified(int item)
```

```
{
  struct node *ptr = (struct node *)malloc(sizeof(struct node));
  struct node *temp;
  int i, loc;
  if(ptr == NULL)
  {
     printf("\n OVERFLOW");
  }
  else
  {
     printf("\nEnter the location\n");
     scanf("%d",&loc);
     temp=head;
     for(i=0;i<loc;i++)
     {
        temp = temp->next;
        if(temp == NULL)
        {
           printf("\ncan't insert\n");
           return;
        }
     }
     ptr->data = item;
     ptr->next = temp->next;
     ptr -> prev = temp;
     temp->next = ptr;
     temp->next->prev=ptr;
     printf("Node Inserted\n");
  }
}
```

```c
#include<stdio.h>
#include<stdlib.h>
void insert_specified(int);
void create(int);
struct node
{
    int data;
    struct node *next;
    struct node *prev;
};
struct node *head;
void main ()
{
    int choice,item,loc;
    do
    {
        printf("\nEnter the item which you want to insert?\n");
        scanf("%d",&item);
        if(head == NULL)
        {
            create(item);
        }
        else
        {
            insert_specified(item);
        }
        printf("\nPress 0 to insert more ?\n");
        scanf("%d",&choice);
    }while(choice == 0);
}
void create(int item)
    {
   struct node *ptr = (struct node *)malloc(sizeof(struct node));
```

```
34      if(ptr == NULL)
35      {
36          printf("\nOVERFLOW");
37      }
38      else
39      {
40
41
42      if(head==NULL)
43      {
44          ptr->next = NULL;
45          ptr->prev=NULL;
46          ptr->data=item;
47          head=ptr;
48      }
49      else
50      {
51          ptr->data=item;printf("\nPress 0 to insert more ?\n");
52          ptr->prev=NULL;
53          ptr->next = head;
54          head->prev=ptr;
55          head=ptr;
56      }
57       printf("\nNode Inserted\n");
58  }
59
60  }
61  void insert_specified(int item)
62  {
63      struct node *ptr = (struct node *)malloc(sizeof(struct node));
64      struct node *temp;
65      int i, loc;
66      if(ptr == NULL)
```

```
    {
        printf("\n OVERFLOW");
    }
    else
    {
        printf("\nEnter the location\n");
        scanf("%d",&loc);
        temp=head;
        for(i=0;i<loc;i++)
        {
            temp = temp->next;
            if(temp == NULL)
            {
                printf("\ncan't insert\n");
                return;
            }
        }
        ptr->data = item;
        ptr->next = temp->next;
        ptr -> prev = temp;
        temp->next = ptr;
        temp->next->prev=ptr;
        printf("Node Inserted\n");
    }
}
```

DELETION :

A) Deletion at beginning

```
#include<stdio.h>
#include<stdlib.h>
void create(int);
void beginning_delete();
struct node
{
   int data;
   struct node *next;
   struct node *prev;
};
struct node *head;
```

```
void main ()
{
   int choice,item;
   do
   {
      printf("1.Append List\n2.Delete node from beginning\n3.Exit\n4.Enter your choice?");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1:
         printf("\nEnter the item\n");
         scanf("%d",&item);
         create(item);
         break;
         case 2:
         beginning_delete();
         break;
         case 3:
         exit(0);
         break;
         default:
         printf("\nPlease enter valid choice\n");
      }

   }while(choice != 3);
}
void create(int item)
{

  struct node *ptr = (struct node *)malloc(sizeof(struct node));
  if(ptr == NULL)
  {
     printf("\nOVERFLOW\n");
  }
  else
  {


  if(head==NULL)
  {
```

```
      ptr->next = NULL;
      ptr->prev=NULL;
      ptr->data=item;
      head=ptr;
    }
    else
    {
      ptr->data=item;printf("\nPress 0 to insert more ?\n");
      ptr->prev=NULL;
      ptr->next = head;
      head->prev=ptr;
      head=ptr;
    }
    printf("\nNode Inserted\n");
}
}
void beginning_delete()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\n UNDERFLOW\n");
    }
    else if(head->next == NULL)
    {
        head = NULL;
        free(head);
        printf("\nNode Deleted\n");
    }
    else
    {
        ptr = head;
        head = head -> next;
        head -> prev = NULL;
        free(ptr);
        printf("\nNode Deleted\n");
    }
}
```

```c
#include<stdio.h>
#include<stdlib.h>
void create(int);
void beginning_delete();
struct node
{
    int data;
    struct node *next;
    struct node *prev;
};
struct node *head;
void main ()
{
    int choice,item;
    do
    {
        printf("1.Append List\n2.Delete node from beginning\n3.Exit\n4.Enter your choice?");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            printf("\nEnter the item\n");
            scanf("%d",&item);
            create(item);
            break;
            case 2:
            beginning_delete();
            break;
            case 3:
            exit(0);
            break;
            default:
            printf("\nPlease enter valid choice\n");
        }

    }while(choice != 3);
}
void create(int item)
{

   struct node *ptr = (struct node *)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
       printf("\nOVERFLOW\n");
   }
```

```
    else
    {


    if(head==NULL)
    {
        ptr->next = NULL;
        ptr->prev=NULL;
        ptr->data=item;
        head=ptr;
    }
    else
    {
        ptr->data=item;printf("\nPress 0 to insert more ?\n");
        ptr->prev=NULL;
        ptr->next = head;
        head->prev=ptr;
        head=ptr;
    }
     printf("\nNode Inserted\n");
}
}
void beginning_delete()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\n UNDERFLOW\n");
    }
    else if(head->next == NULL)
    {
        head = NULL;
        free(head);
          printf("\nNode Deleted\n");
     }
     else
     {
         ptr = head;
         head = head -> next;
         head -> prev = NULL;
         free(ptr);
         printf("\nNode Deleted\n");
     }
}
```

B) Deletion in doubly linked list at the end

```
#include<stdio.h>
#include<stdlib.h>
void create(int);
void last_delete();
struct node
{
   int data;
   struct node *next;
   struct node *prev;
};
struct node *head;
void main ()
{
   int choice,item;
   do
   {
      printf("1.Append List\n2.Delete node from end\n3.Exit\n4.Enter your choice?");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1:
         printf("\nEnter the item\n");
         scanf("%d",&item);
         create(item);
         break;
         case 2:
         last_delete();
         break;
         case 3:
         exit(0);
         break;
         default:
         printf("\nPlease enter valid choice\n");
      }

   }while(choice != 3);
}
void create(int item)
```

```
{

  struct node *ptr = (struct node *)malloc(sizeof(struct node));
  if(ptr == NULL)
  {
     printf("\nOVERFLOW\n");
  }
  else
  {


  if(head==NULL)
  {
     ptr->next = NULL;
     ptr->prev=NULL;
     ptr->data=item;
     head=ptr;
  }
  else
  {
     ptr->data=item;
     ptr->prev=NULL;
     ptr->next = head;
     head->prev=ptr;
     head=ptr;
  }
   printf("\nNode Inserted\n");
}
}
void last_delete()
{
   struct node *ptr;
   if(head == NULL)
   {
      printf("\n UNDERFLOW\n");
   }
   else if(head->next == NULL)
   {
      head = NULL;
      free(head);
```

```
        printf("\nNode Deleted\n");
    }
    else
    {
        ptr = head;
        if(ptr->next != NULL)
        {
            ptr = ptr -> next;
        }
        ptr -> prev -> next = NULL;
        free(ptr);
        printf("\nNode Deleted\n");
    }
}
```

```c
#include<stdio.h>
#include<stdlib.h>
void create(int);
void last_delete();
struct node
{
    int data;
    struct node *next;
    struct node *prev;
};
struct node *head;
void main ()
{
    int choice,item;
    do
    {
        printf("1.Append List\n2.Delete node from end\n3.Exit\n4.Enter your choice?");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            printf("\nEnter the item\n");
            scanf("%d",&item);
            create(item);
            break;
            case 2:
            last_delete();
            break;
            case 3:
            exit(0);
            break;
            default:
            printf("\nPlease enter valid choice\n");
        }

    }while(choice != 3);
}
void create(int item)
{

   struct node *ptr = (struct node *)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
       printf("\nOVERFLOW\n");
   }
   else
   {
```

```
    if(head==NULL)
    {
        ptr->next = NULL;
        ptr->prev=NULL;
        ptr->data=item;
        head=ptr;
    }
    else
    {
        ptr->data=item;
        ptr->prev=NULL;
        ptr->next = head;
        head->prev=ptr;
        head=ptr;
    }
     printf("\nNode Inserted\n");
}
}
void last_delete()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\n UNDERFLOW\n");
    }
    else if(head->next == NULL)
    {
        head = NULL;
        free(head);
        printf("\nNode Deleted\n");
    }
    else
    {
        ptr = head;
        if(ptr->next != NULL)
        {
            ptr = ptr -> next;
        }
        ptr -> prev -> next = NULL;
        free(ptr);
        printf("\nNode Deleted\n");
    }
}
```

C) Deletion in doubly linked list after the specified node

```
#include<stdio.h>
#include<stdlib.h>
void create(int);
void delete_specified();
struct node
{
   int data;
   struct node *next;
   struct node *prev;
};
struct node *head;
void main ()
{
   int choice,item;
   do
   {
      printf("1.Append List\n2.Delete node\n3.Exit\n4.Enter your choice?");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1:
         printf("\nEnter the item\n");
         scanf("%d",&item);
         create(item);
         break;
         case 2:
         delete_specified();
         break;
         case 3:
         exit(0);
         break;
         default:
         printf("\nPlease enter valid choice\n");
      }

   }while(choice != 3);
}
void create(int item)
{
  struct node *ptr = (struct node *)malloc(sizeof(struct node));
```

```
  if(ptr == NULL)
  {
     printf("\nOVERFLOW\n");
  }
  else
  {


  if(head==NULL)
  {
     ptr->next = NULL;
     ptr->prev=NULL;
     ptr->data=item;
     head=ptr;
  }
  else
  {
     ptr->data=item;
     ptr->prev=NULL;
     ptr->next = head;
     head->prev=ptr;
     head=ptr;
  }
   printf("\nNode Inserted\n");
}

}
void delete_specified( )
{
   struct node *ptr, *temp;
   int val;
   printf("Enter the value");
   scanf("%d",&val);
   temp = head;
   while(temp -> data != val)
   temp = temp -> next;
   if(temp -> next == NULL)
   {
      printf("\nCan't delete\n");
   }
```

```
    else if(temp -> next -> next == NULL)
    {
        temp ->next = NULL;
        printf("\nNode Deleted\n");
    }
    else
    {
        ptr = temp -> next;
        temp -> next = ptr -> next;
        ptr -> next -> prev = temp;
        free(ptr);
        printf("\nNode Deleted\n");
    }
}
```

```c
#include<stdio.h>
#include<stdlib.h>
void create(int);
void delete_specified();
struct node
{
    int data;
    struct node *next;
    struct node *prev;
};
struct node *head;
void main ()
{
    int choice,item;
    do
    {
        printf("1.Append List\n2.Delete node\n3.Exit\n4.Enter your choice?");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            printf("\nEnter the item\n");
            scanf("%d",&item);
            create(item);
            break;
            case 2:
            delete_specified();
            break;
            case 3:
            exit(0);
            break;
            default:
            printf("\nPlease enter valid choice\n");
        }

    }while(choice != 3);
}
void create(int item)
{
   struct node *ptr = (struct node *)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
       printf("\nOVERFLOW\n");
   }
   else
   {
   if(head==NULL)
```

```
    {
        ptr->next = NULL;
        ptr->prev=NULL;
        ptr->data=item;
        head=ptr;
    }
    else
    {
        ptr->data=item;
        ptr->prev=NULL;
        ptr->next = head;
        head->prev=ptr;
        head=ptr;
    }
     printf("\nNode Inserted\n");
}

}
void delete_specified( )
{
    struct node *ptr, *temp;
    int val;
    printf("Enter the value");
    scanf("%d",&val);
    temp = head;
    while(temp -> data != val)
    temp = temp -> next;
    if(temp -> next == NULL)
    {
        printf("\nCan't delete\n");
    }
    else if(temp -> next -> next == NULL)
    {
        temp ->next = NULL;
        printf("\nNode Deleted\n");
    }
    else
    {
        ptr = temp -> next;
        temp -> next = ptr -> next;
        ptr -> next -> prev = temp;
        free(ptr);
        printf("\nNode Deleted\n");
    }
}
```

Circular Linked List

Insertion:-

A) Insertion at beginning

```
#include<stdio.h>
#include<stdlib.h>
void beg_insert(int);
struct node
{
   int data;
   struct node *next;
};
struct node *head;
void main ()
{
   int choice,item;
   do
   {
      printf("\nEnter the item which you want to insert?\n");
      scanf("%d",&item);
      beg_insert(item);
      printf("\nPress 0 to insert more ?\n");
      scanf("%d",&choice);
   }while(choice == 0);
}
void beg_insert(int item)
{

   struct node *ptr = (struct node *)malloc(sizeof(struct node));
   struct node *temp;
   if(ptr == NULL)
   {
      printf("\nOVERFLOW");
   }
   else
   {
      ptr -> data = item;
      if(head == NULL)
      {
         head = ptr;
         ptr -> next = head;
      }
      else
```

```
    {
        temp = head;
        while(temp->next != head)
            temp = temp->next;
        ptr->next = head;
        temp -> next = ptr;
        head = ptr;
    }
  printf("\nNode Inserted\n");
  }

}
```

```c
#include<stdio.h>
#include<stdlib.h>
void beg_insert(int);
struct node
{
    int data;
    struct node *next;
};
struct node *head;
void main ()
{
    int choice,item;
    do
    {
        printf("\nEnter the item which you want to insert?\n");
        scanf("%d",&item);
        beg_insert(item);
        printf("\nPress 0 to insert more ?\n");
        scanf("%d",&choice);
    }while(choice == 0);
}
void beg_insert(int item)
{

    struct node *ptr = (struct node *)malloc(sizeof(struct node));
    struct node *temp;
    if(ptr == NULL)
    {
        printf("\nOVERFLOW");
    }
    else
    {
        ptr -> data = item;
```

```
        {
            head = ptr;
            ptr -> next = head;
        }
        else
        {
            temp = head;
            while(temp->next != head)
                temp = temp->next;
            ptr->next = head;
            temp -> next = ptr;
            head = ptr;
        }
    printf("\nNode Inserted\n");
    }
}
```

B) Insertion at the end
FUNCTION

```
void lastinsert(struct node*ptr, struct node *temp, int item)
{
   ptr = (struct node *)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
      printf("\nOVERFLOW\n");
   }
   else
   {
      ptr->data = item;
      if(head == NULL)
      {
         head = ptr;
         ptr -> next = head;
      }
      else
      {
         temp = head;
         while(temp -> next != head)
         {
            temp = temp -> next;
         }
         temp -> next = ptr;
         ptr -> next = head;
```

```
    }
  }

}
```

```c
void lastinsert(struct node*ptr, struct node *temp, int item)
{
    ptr = (struct node *)malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW\n");
    }
    else
    {
        ptr->data = item;
        if(head == NULL)
        {
            head = ptr;
            ptr -> next = head;
        }
        else
        {
            temp = head;
            while(temp -> next != head)
            {
                temp = temp -> next;
            }
            temp -> next = ptr;
            ptr -> next = head;
        }
    }

}
```

DELETION:-

A) Deletion in circular singly linked list at beginning

nclude<stdio.h>

#include<stdlib.h>

```
void create(int);
void beg_delete();
struct node
{
    int data;
    struct node *next;
};
struct node *head;
void main ()
{
    int choice,item;
    do
    {
        printf("1.Append List\n2.Delete Node from beginning\n3.Exit\n4.Enter your choice?");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            printf("\nEnter the item\n");
            scanf("%d",&item);
            create(item);
```

```
            break;
            case 2:
            beg_delete();
            break;
            case 3:
            exit(0);
            break;
            default:
            printf("\nPlease Enter valid choice\n");
        }

    }while(choice != 3);
}
void create(int item)
{

    struct node *ptr = (struct node *)malloc(sizeof(struct node));
    struct node *temp;
    if(ptr == NULL)
    {
        printf("\nOVERFLOW");
```

```
}
else
{
    ptr -> data = item;
    if(head == NULL)
    {
        head = ptr;
        ptr -> next = head;
    }
    else
    {
        temp = head;
        while(temp->next != head)
            temp = temp->next;
        ptr->next = head;
        temp -> next = ptr;
        head = ptr;
    }
printf("\nNode Inserted\n");
}
```

```
}
void beg_delete()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\nUNDERFLOW\n");
    }
    else if(head->next == head)
    {
        head = NULL;
        free(head);
        printf("\nNode Deleted\n");
    }

    else
    {
        ptr = head;
        while(ptr -> next != head)
            ptr = ptr -> next;
        ptr->next = head->next;
```

```
        free(head);
        head = ptr->next;
        printf("\nNode Deleted\n");
    }
}
```

```c
#include<stdio.h>
#include<stdlib.h>
void create(int);
void beg_delete();
struct node
{
    int data;
    struct node *next;
};
struct node *head;
void main ()
{
    int choice,item;
    do
    {
        printf("1.Append List\n2.Delete Node from beginning\n3.Exit\n4.Enter your choice?");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            printf("\nEnter the item\n");
            scanf("%d",&item);
            create(item);
            break;
            case 2:
            beg_delete();
            break;
            case 3:
            exit(0);
            break;
            default:
            printf("\nPlease Enter valid choice\n");
        }
    }while(choice != 3);
}
void create(int item)
{
    struct node *ptr = (struct node *)malloc(sizeof(struct node));
    struct node *temp;
    if(ptr == NULL)
    {
        printf("\nOVERFLOW");
    }
    else
    {
        ptr -> data = item;
        if(head == NULL)
        {
```

```
            head = ptr;
            ptr -> next = head;
        }
        else
        {
            temp = head;
            while(temp->next != head)
                temp = temp->next;
            ptr->next = head;
            temp -> next = ptr;
            head = ptr;
        }
    printf("\nNode Inserted\n");
    }
}
void beg_delete()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\nUNDERFLOW\n");
    }
    else if(head->next == head)
    {
        head = NULL;
        free(head);
        printf("\nNode Deleted\n");
    }
    else
    {
        ptr = head;
        while(ptr -> next != head)
            ptr = ptr -> next;
```

```
            ptr->next = head->next;
            free(head);
            head = ptr->next;
            printf("\nNode Deleted\n");
        }
}
```

B) Deletion in Circular singly linked list at the end

#include<stdio.h>

#include<stdlib.h>

```
void create(int);
void last_delete();
struct node
{
    int data;
    struct node *next;
};
struct node *head;
void main ()
{
    int choice,item;
    do
    {
        printf("1.Append List\n2.Delete Node from end\n3.Exit\n4.Enter your choice?");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            printf("\nEnter the item\n");
            scanf("%d",&item);
            create(item);
```

```
            break;
            case 2:
            last_delete();
            break;
            case 3:
            exit(0);
            break;
            default:
            printf("\nPlease Enter valid choice\n");
        }

    }while(choice != 3);
}
void create(int item)
{
    struct node *ptr = (struct node *)malloc(sizeof(struct node));
    struct node *temp;
    if(ptr == NULL)
    {
        printf("\nOVERFLOW\n");
    }
```

```
else
{
    ptr -> data = item;
    if(head == NULL)
    {
        head = ptr;
        ptr -> next = head;
    }
    else
    {
        temp = head;
        while(temp->next != head)
            temp = temp->next;
        ptr->next = head;
        temp -> next = ptr;
        head = ptr;
    }
printf("\nNode Inserted\n");
}

}
```

```
void last_delete()
{
struct node *ptr, *preptr;
    if(head==NULL)
    {
        printf("\nUNDERFLOW\n");
    }
    else if (head ->next == head)
    {
        head = NULL;
        free(head);
        printf("\nNode Deleted\n");
    }
    else
    {
        ptr = head;
        while(ptr ->next != head)
        {
            preptr=ptr;
            ptr = ptr->next;
        }
```

```
        preptr->next = ptr -> next;
        free(ptr);
        printf("\nNode Deleted\n");
    }
}
```

5. Observations/Discussions/ Complexity Analysis:

In a doubly-linked list, the time complexity for inserting and deleting an element is **O(1)**.

Circular Linked List Complexity

Circular Linked List Complexity	**Time Complexity**	**Space Complexity**
Insertion Operation	O(1) or O(n)	O(1)
Deletion Operation	O(1)	O(1)

6. Result/Output/Writing Summary:-

Doubly Linked List

INSERTION:-

A) Insertion in doubly linked list at beginning

```
Enter the item which you want to insert?
5

Press 0 to insert more ?
0

Enter the item which you want to insert?
24

Press 0 to insert more ?
2


...Program finished with exit code 0
Press ENTER to exit console.
```

B) Insertion in doubly linked list at the end

```
Enter the item which you want to insert?
35

Node Inserted

Press 0 to insert more ?
2


...Program finished with exit code 0
Press ENTER to exit console.
```

C) Insertion in doubly linked list after Specified node

```
Enter the item which you want to insert?
23

Node Inserted

Press 0 to insert more ?
2


...Program finished with exit code 0
Press ENTER to exit console.
```

DELETION :

A) Deletion at beginning

```
1.Append List
2.Delete node from beginning
3.Exit
4.Enter your choice?1

Enter the item
12

Node Inserted
1.Append List
2.Delete node from beginning
3.Exit
4.Enter your choice?2

Node Deleted
1.Append List
2.Delete node from beginning
3.Exit
4.Enter your choice?3


...Program finished with exit code 0
Press ENTER to exit console.
```

B) Deletion in doubly linked list at the end

```
1.Append List
2.Delete node from end
3.Exit
4.Enter your choice?1

Enter the item
23

Node Inserted
1.Append List
2.Delete node from end
3.Exit
4.Enter your choice?2

Node Deleted
1.Append List
2.Delete node from end
3.Exit
4.Enter your choice?3


...Program finished with exit code 0
Press ENTER to exit console.
```

C) Deletion in doubly linked list after the specified node

```
1.Append List
2.Delete node
3.Exit
4.Enter your choice?1

Enter the item
10

Node Inserted
1.Append List
2.Delete node
3.Exit
4.Enter your choice?1

Enter the item
20

Node Inserted
1.Append List
2.Delete node
3.Exit
4.Enter your choice?1

Enter the item
30

Node Inserted
1.Append List
2.Delete node
3.Exit
4.Enter your choice?2
Enter the value20

Node Deleted
1.Append List
2.Delete node
3.Exit
4.Enter your choice?3


...Program finished with exit code 0
Press ENTER to exit console.
```

Circular Linked List

Insertion:-

A) Insertion at beginning

```
Enter the item which you want to insert?
23

Node Inserted

Press 0 to insert more ?
2


...Program finished with exit code 0
Press ENTER to exit console.
```

DELETION:-

A) Deletion in circular singly linked list at beginning

```
1.Append List
2.Delete Node from beginning
3.Exit
4.Enter your choice?1

Enter the item
34

Node Inserted
1.Append List
2.Delete Node from beginning
3.Exit
4.Enter your choice?2

Node Deleted
1.Append List
2.Delete Node from beginning
3.Exit
4.Enter your choice?3


...Program finished with exit code 0
Press ENTER to exit console.
```

B) Deletion in Circular singly linked list at the end

```
1.Append List
2.Delete Node from end
3.Exit
4.Enter your choice?1

Enter the item
34

Node Inserted
1.Append List
2.Delete Node from end
3.Exit
4.Enter your choice?2

Node Deleted
1.Append List
2.Delete Node from end
3.Exit
4.Enter your choice?3


...Program finished with exit code 0
Press ENTER to exit console.
```

Learning outcomes (What I have learnt):

1. Linked list.
2. Types of Linked List.
3. Operation .

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.			
2.			
3.			