Program

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

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

struct node *newnode,*head=NULL,*temp;

void insert_beg(int item)
{
    newnode=(struct node*)malloc(sizeof(struct node));
    if(newnode==NULL)
    {
        printf("Insufficient memory");
    }
    else if(head==NULL)
    {
        head=newnode;
        newnode->data=item;
        newnode->next=NULL;
    }
    else
    {
        newnode->data=item;
        newnode->next=head;
        head=newnode;
    }
}

void insert_end(int item)
{
    newnode=(struct node*)malloc(sizeof(struct node));
    if(newnode==NULL)
    {
        printf("Insufficient memory");
    }
    else if(head==NULL)
    {
        head=newnode;
        newnode->data=item;
        newnode->next=NULL;
    }
    else
    {
        newnode->data=item;
        temp=head;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=newnode;
    }
}

void insert_atany(int item, int pos)
{
    newnode=(struct node*)malloc(sizeof(struct node));
    int count=0;
```

Experiment No: 12
Date:

SINGLY LINKED LIST

Aim:

To write a C program to create, insert, delete and display nodes in a singly linked list.

Algorithm:

1. Start
2. Define Struct node

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

3. Create node pointers newnode, head and temp
4. Function to create newnode

```
newnode = (struct node*)malloc(sizeof(struct node))
printf("Enter the data:")
scanf("%d",&newnode->data)
newnode ->next = NULL
```

5. Function to insert at front

```
newnode=(struct node*)malloc(sizeof(struct node))
If newnode is NULL:
    print "Insufficient Memory"
else if head is NULL:
    head = newnode
    newnode->data=item
    newnode->next = NULL
else
    newnode->date=item
    newnode->next = NULL
    head=newnode
    temp=head;
    while(temp!=NULL)
    {
        temp=temp->next;
        count++;
    }
    if(pos>count+1 || pos<1)
    {
        printf("Invalid choice");
    }
    else
    {
    temp=head;
    for(int i=1;i<pos;i++)
    {
        temp=temp->next;
    }
    newnode->data=item;
    newnode->next=temp->next;
    temp->next=newnode;
    }
}

void delete_beg()
{
    if(head==NULL)
    {
        printf("list is empty");
    }
    else
    {
        struct node*current=head;
        head=head->next;
        free(current);
    }
}

void delete_end()
{
    struct node*prev;
    if(head==NULL)
    {
        printf("List is empty");
    }
    else
    {
      temp=head;
      while(temp->next!=NULL)
      {
          prev=temp;
          temp=temp->next;
      }
      prev->next=NULL;
      free(temp);
    }
}

void delete_atany(int pos)
{
    int count=0;
    struct node*temp=head,*prev;
    while(temp!=NULL)
    {
```

6. Function to insert at end

```
newnode=(struct node*)malloc(sizeof(struct node))
if newnode is NULL:
    print "Insufficient Memory"
else if head is NULL:
    head = newnode
    newnode->data=item
    newnode->next = NULL
else
    newnode->data=item
    temp=head
    While temp->next!=NULL
        temp = temp->next
        End while
    temp->next = newnode
```

7. Function to create at any position

```
newnode=(struct node*)malloc(sizeof(struct node))
int count=0
while temp!=NULL
    temp = temp->next
    count++
end while
temp = head
if pos>count+1 or pos<1
    print 'Invalid position'
else
    temp=head
    for i=1 to pos
        temp = temp->next
    end for
    newnode->data = item
    newnode->next = temp->next
    temp->next = newnode
End if
```

8. Function to delete from beginning

```
if head is NULL
    print "List is empty"
else
    temp = head
    while temp->next!=NULL
        prev=temp
        temp=temp->next
    End while
    prev->next=NULL
    free(temp)
```

```
        temp=temp->next;
        count++;
    }
    if(pos>count || pos<1)
    {
        printf("invalid position");
    }
    temp=head;
    for(int i=1;i<pos;i++)
    {
        prev=temp;
        temp=temp->next;
    }
    prev->next=temp->next;
    free(temp);
}

void display()
{
    temp=head;
    while(temp!=NULL)
    {
        printf("%d",temp->data);
        temp=temp->next;
    }
}

void main()
{
    int ch,item,pos;
    char y;
    do{
    printf("\n1. Enqueue at beginning");
    printf("\n2. Enqueue at end");
    printf("\n3. Enqueue at any position");
    printf("\n4. Delete from beginning");
    printf("\n5. Delete from end");
    printf("\n6. Delete from any position");
    printf("\n7. Display");
    printf("\nEnter your choice:");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1:
        {
          printf("Enter the item to insert:");
          scanf("%d",&item);
          insert_beg(item);
          break;
        }
        case 2:
        {
          printf("Enter the item to insert:");
          scanf("%d",&item);
          insert_end(item);
          break;
        }
        case 3:
        {
          printf("Enter the item to insert:");
          scanf("%d",&item);
          printf("Enter position to insert at:");
          scanf("%d",&pos);
```

9. Function to delete from end

```
struct node * prev
if head is NULL
    print 'List empty'
else
    temp=head
    while temp->next!=NULL
        prev = temp
        temp=temp->next
    end while
    prev->next = NULL
    free(temp)
```

10. Function to delete from any position

```
struct node*temp = head,*prev
int count =0
while temp!=NULL
    count++
    temp=temp->next
end while
if pos>count+1 or pos<1
    print 'Invalid position'
else
    temp = head
    for i=1 to pos
        prev = temp
        temp = temp->next
    end for
    prev->next = temp->next
    free(temp)
```

11. Function to display elements

```
temp = head
while temp!= NULL
    print temp->data
    temp = temp->next
end while
```

12. Create main function

```
int ch,item,pos
char y
do:
    print "Enqueue at beginning"
    print "Enqueue at end"
    print "Enqueue at any posotion"
    print "Delete from beginning"
    print "Delete from end"
    print "Delete from any position"
    print "Display"
    print "Enter your choice:"
    scan to ch
    start switch(ch)
        case 1:
            print "Enter the item to insert:"
            scan to item
            call insert_beg(item)
```

```
          insert_atany(item,pos);
          break;
        }
        case 4:
        {
          delete_beg();
          break;
        }
        case 5:
        {
            delete_end();
            break;
        }
        case 6:
        {
            printf("Enter position :");
            scanf("%d",&pos);
            delete_atany(pos);
            break;
        }
        case 7:
        {
            display();
            break;
        }
        default:
        {
            printf("Invalid choice");
        }
    }
    printf("\nDo u want to continue:");
    scanf("%s",&y);
  }while(y=='Y'||y=='y');
}
```

```
                break
            case 2:
                print "Enter the item to insert:"
                scan to item
                call insert_end(item)
                break
            case 3:
                print "Enter the item to insert:"
                scan to item
                print "Enter the position to insert at:"
                scan to pos
                call insert_atany(item,pos)
                break
            case 4:
                call delete_beg()
                break
            case 5:
                call delete_end()
                break
            case 6:
                 print "Enter position :"
                scan to pos
                call delete_atany(pos)
                break
            case 7:
                call display()
                break
            default:
                print "Invalid choice"
                end switch

        print "Do u want to continue:"
        scan to y
    continue while still y=='y' or y=='Y'
```

13. Stop

Output

```
1. Enqueue at beginning
2. Enqueue at end
3. Enqueue at any position
4. Delete from beginning
5. Delete from end
6. Delete from any position
7. Display
Enter your choice:1
Enter the item to insert:2
Do u want to continue:y

1. Enqueue at beginning
2. Enqueue at end
3. Enqueue at any position
4. Delete from beginning
5. Delete from end
6. Delete from any position
7. Display
Enter your choice:1
Enter the item to insert:1
Do u want to continue:y

1. Enqueue at beginning
2. Enqueue at end
3. Enqueue at any position
4. Delete from beginning
5. Delete from end
6. Delete from any position
7. Display
Enter your choice:2
Enter the item to insert:4
Do u want to continue:y

1. Enqueue at beginning
2. Enqueue at end
3. Enqueue at any position
4. Delete from beginning
5. Delete from end
6. Delete from any position
7. Display
Enter your choice:7
124
Do u want to continue:y

1. Enqueue at beginning
2. Enqueue at end
3. Enqueue at any position
4. Delete from beginning
5. Delete from end
6. Delete from any position
7. Display
Enter your choice:3
Enter the item to insert:3
Enter position to insert at:3
Do u want to continue:y

1. Enqueue at beginning
2. Enqueue at end
3. Enqueue at any position
4. Delete from beginning
```

```
5. Delete from end
6. Delete from any position
7. Display
Enter your choice:7
1234
Do u want to continue:y

1. Enqueue at beginning
2. Enqueue at end
3. Enqueue at any position
4. Delete from beginning
5. Delete from end
6. Delete from any position
7. Display
Enter your choice:4
Do u want to continue:y

1. Enqueue at beginning
2. Enqueue at end
3. Enqueue at any position
4. Delete from beginning
5. Delete from end
6. Delete from any position
7. Display
Enter your choice:5
Do u want to continue:y

1. Enqueue at beginning
2. Enqueue at end
3. Enqueue at any position
4. Delete from beginning
5. Delete from end
6. Delete from any position
7. Display
Enter your choice:6
Enter position :2
Do u want to continue:y

1. Enqueue at beginning
2. Enqueue at end
3. Enqueue at any position
4. Delete from beginning
5. Delete from end
6. Delete from any position
7. Display
Enter your choice:4
Do u want to continue:y

1. Enqueue at beginning
2. Enqueue at end
3. Enqueue at any position
4. Delete from beginning
5. Delete from end
6. Delete from any position
7. Display
Enter your choice:4
list is empty
Do u want to continue:n
```

Result:

Program has been executed successfully and output has been obtained.