```
//PROGRAM FOR SINGLE LINKED LIST(SLL)
//CREATION,INSERTION,DELETION AND TRAVERSING
#include<stdio.h>
#include<conio.h>
#include<process.h>
struct node
{
int info;
node *next;
}*head,*current,*temp;

void insert()
{
int x;
printf("Enter the value");
scanf("%d",&x);
temp= (struct node*)malloc(sizeof(struct node));
temp->info=x;
temp->next=NULL;
        if(head==NULL)
        {
        head=temp;
        current=temp;
        }
        else
        {
        current->next=temp;
        current=current->next;
        }
}
void display()
{
        temp=head;
        while(temp!=NULL)
        {
        printf("%d",temp->info);
        temp=temp->next;
        }
}
void main()
{
int c=-1;
while(c!=0)
{
printf("\nEnter the choice");
printf("\n1. INSERT NODE");
printf("\n2. DELETE NODE");
printf("\n3. DISPLAY LIST");
printf("\n0. EXIT");
scanf("%d",&c);

        switch(c)
        {
        case 1: insert();
                break;
        case 2: delete();
```

WRITE AN ALGORITHM TO CREATE SINGLE LINKED LIST.
OR
WRITE AN ALGORITHM TO INSERT ELEMENTS INTO SLL

ASSUMPTION:

(a) Let head is pointer to first node of SLL
(b) Let current is pointing to last node of SLL
(c) Declarion of node

```
        struct node
        {
        int info;
        node *next;
        }*head,*current,*temp;
```

(d) Initialization

```
        head=current=temp=NULL
```

STEPS FOR ALGORITHM:

```
Step-1 : Read element and store in x
Step-2 : Creat new node pointed by temp
         temp= (struct
node*)malloc(sizeof(struct node));
Step-3 : Assign x to temp node
         temp->info=x;
         temp->next=NULL;
Step-4: Insert temp into SLL
        if(head==NULL)
        {
        head=temp;
        current=temp;
        }
        else
```

```
WRITE AN ALGORITHM TO SEARCH A NODE IN SINGLE LINKED LIST.
ASSUMPTION:
Declarion of node
        struct node
        {
        int info;
        node *next;
        }*head,*current;

STEPS FOR ALGORITHM:

Step-1 : Read element X to search
Step-2 : current=head
Step-3 : FOUND = false
Step-3 : while(current!=null)
         {
                if(current->info=X)
                {
                FOUND=true;
                goto step-4;
                }
          current=current->next
         }

Step-4 : if(FOUND=true)
         PRINT "ELEMENT FOUND"
         else
         PRINT "ELEMENT NOT FOUND"
Step-5 : STOP.
```

WRITE AN ALGORITHM TO DELETE A NODE FROM SINGLE LINKED LIST.

ASSUMPTION:

Declarion of node

```
        struct node
        {
        int info;
        node *next;
        }*current;
```

STEPS FOR ALGORITHM:

```
Step-1 : Read element/node to delete
Step-2 : Search element in the SLL to be deleted.
Step-3 : if element is found and let it be current->next
         then
         current->next=current->next->next
         else
         PRINT "Element not found"
Step-4 : STOP.
```

```
WRITE AN ALGORITHM TO TRAVERSING SINGLE LINKED LIST.

ASSUMPTION:
Declarion of node
        struct node
        {
        int info;
        node *next;
        }*head,*current;

STEPS FOR ALGORITHM:

Step-1 : Set current=head
Step-2 : while(current!=null)
         {
         PRINT current->info
         current=current->next
         }
Step-3 : STOP.
```

WRITE AN ALGORITHM TO delete tail(last element)
of SINGLE LINKED LIST.

ASSUMPTION:
Declarion of node

```
struct node
{
int info;
node *next;
}*head,*tail,*temp;
```

(a) let head be the first node(head)
(b) let tail be the last node(tail)
(c) let temp be temporary pointer node

STEPS FOR ALGORITHM:

Step-1 : Move temp to second last node
Step-2 : temp->next =tail
Step-3 : temp->next =NULL
Step-4 : tail=temp
Step-5 : STOP

WRITE AN ALGORITHM TO A INSERT NODE IN SINGLE LINKED LIST.

ASSUMPTION:
Declarion of node

```
        struct node
        {
        int info;
        node *next;
        }*head,*current,*temp;
```

(a) let head be the first node(head)
(b) let current be the node after which element to be inserted
(c) let temp be temporary pointer node

STEPS FOR ALGORITHM:

```
Step-1 : Create temp node to be inserted
Step-2:  let temp will be inserted after current node
Step-3 : temp->next = current->next
         current->next = temp
Step-4:  STOP
```

WRITE AN ALGORITHM TO DELETE A NODE FROM SINGLE LINKED LIST.
ASSUMPTION:
Declarion of node

```
        struct node
        {
        int info;
        node *next;
        }*current;
```

STEPS FOR ALGORITHM:

```
Step-1 : Read element/node to delete
Step-2 : Search element in the SLL to be deleted.
Step-3 : if element is found and let it be current->next
         then
         current->next=current->next->next
         else
         PRINT "Element not found"
Step-4 : STOP.

                OR
Step-1 : Read element X to delete
Step-2 : current->next= Search(X)
Step-3 : if (current->next!=null)
         then
         current->next=current->next->next
         else
         PRINT "Element not found"
Step-4 : STOP.
```

```
WRITE AN ALGORITHM TO SEARCH A NODE IN SINGLE LINKED LIST.
ASSUMPTION:
Declarion of node
        struct node
        {
        int info;
        node *next;
        }*head,*current;

STEPS FOR ALGORITHM:

Step-1 : Read element X to search
Step-2 : current=head
Step-3 : FOUND = false
Step-3 : while(current!=null)
         {
                if(current->info=X)
                {
                FOUND=true;
                goto step-4;
                }
          current=current->next
         }

Step-4 : if(FOUND=true)
         PRINT "ELEMENT FOUND"
         else
         PRINT "ELEMENT NOT FOUND"
Step-5 : STOP.
```