**Linked list**

**-Tanisha Gotadke 1BM21CS229**

1) WAP to Implement Singly Linked List  
with following operations   (10 Marks)  
a)  Create  
a linked list.  
b)   Insertion  
of a node at first position, at any position and at end of list.  
c)   Display  
the contents of the linked list.  
2) WAP to Implement Singly Linked List  
with following operations   (10 Marks)  
a) Create a linked list.  
b) Deletion  
of first element, specified element and last element in the list.  
c)  Display  
the contents of the linked list.

**Code :**

#include<stdio.h>

struct Node

{

int data;

struct Node \*link;

};

typedef struct Node Node;

Node \*start = NULL;

void create()

{

Node \*start,\* curr, \*new;

int ch;

start =(Node\*)malloc(sizeof (Node));

curr = start;

printf("Enter the element\n");

scanf("%d",&new -> data);

while(1)

{

printf("do you want to enter another element (1/0)\n");

scanf("%d",&ch);

if(ch==1)

{

new = (Node\*)malloc(sizeof(Node));

printf("Enter Element");

scanf("%d",&new->data);

curr->link = new;

curr = new;

}

else

{

curr->link = NULL;

break;

}

}

return;

}

void insert\_end()

{

Node \*new, \*temp;

new=(Node\*)malloc(sizeof(Node));

printf("enter element");

scanf("%d",&new->data);

if (start==NULL)

{

start=new;

new->link=NULL;

return;

}

else

{

temp=start;

while(temp->link!=NULL)

{

temp=temp->link;

}

temp->link=new;

new->link=NULL;

return;

}

}

void insert\_pos(int pos)

{

Node \*new, \*temp;

new=(Node\*)malloc(sizeof(Node));

printf("enter element:");

scanf("%d",&new->data);

if(pos==1)

{

new->link=start;

start=new;

return;

}

else

{

temp=start;

int count=1;

while(count<pos)

{

temp=temp->link;

}

new->link=temp->link;

temp->link=new;

return;

}

}

void insert\_beg()

{

Node \* new;

new =(Node\*)malloc(sizeof(Node));

printf("Enter Element\n");

scanf("%d",&new->data);

if(start==NULL)

{

new->link = NULL;

start=new;

return;

}

else

{

new->link = start;

start = new;

return;

}

}

void delete\_beg()

{

Node \*temp;

if (start==NULL)

{

printf("Empty linked list");

return;

}

else

{

temp=start;

start=start->link;

free(temp);

return;

}}

void delete\_end()

{

if(start==NULL)

{

printf("start=empty ll ");

return;

}

else if(start->link==NULL)

{

free(start);

start=NULL;

return;

}

else

{

Node \*previous,\*next;

previous=start;

next=start->link;

while(next->link!=NULL)

{

previous=next;

next=next->link;

}

previous->link=NULL;

free(next);

return;

}

}

void delete\_pos(int del)

{

if(start==NULL)

{

printf("Empty linked list\n");

return;

}

else if(start->data==del)

{

free(start);

start=NULL;

printf("element deleted");

return;

}

else

{

Node \*prev,\*next;

prev=start;

next=start->link;

while(next->data!=del || next!=NULL)

{

prev=next;

next=next->link;

}

if(next->data==del)

{

prev->link=next->link;

free(next);

return;

}

else

{

printf("Element not found");

return;

}

}

}

void display()

{

Node \*temp;

if(start==NULL)

{

printf("Empty linked list\n");

return;

}

temp=start;

while(temp!=NULL)

{

printf("%d",temp->data);

temp=temp->link;

}

}

int main()

{

int choice,x;

printf ("1.create\n 2.insert at beginning\n 3.insert at end\n 4.insert at position\n 5.delete from beginning\n 6.delete from end\n 7.delete from a position\n 8.display\n 9.exit\n");

scanf("%d",&choice);

do

{

switch(choice)

{

case(1): create();

break;

case(2): insert\_beg();

break;

case(3): insert\_end();

break;

case(4): printf("enter the position");

scanf("%d",&x);

insert\_pos(x);

break;

case(5): delete\_beg();

break;

case(6): delete\_end();

break;

case(7): printf("enter the element to delete");

scanf("%d",&x);

delete\_pos(x);

break;

case(8): display();

break;

case(9): exit(0);

default:

printf("enter a valid choice\n");

break;

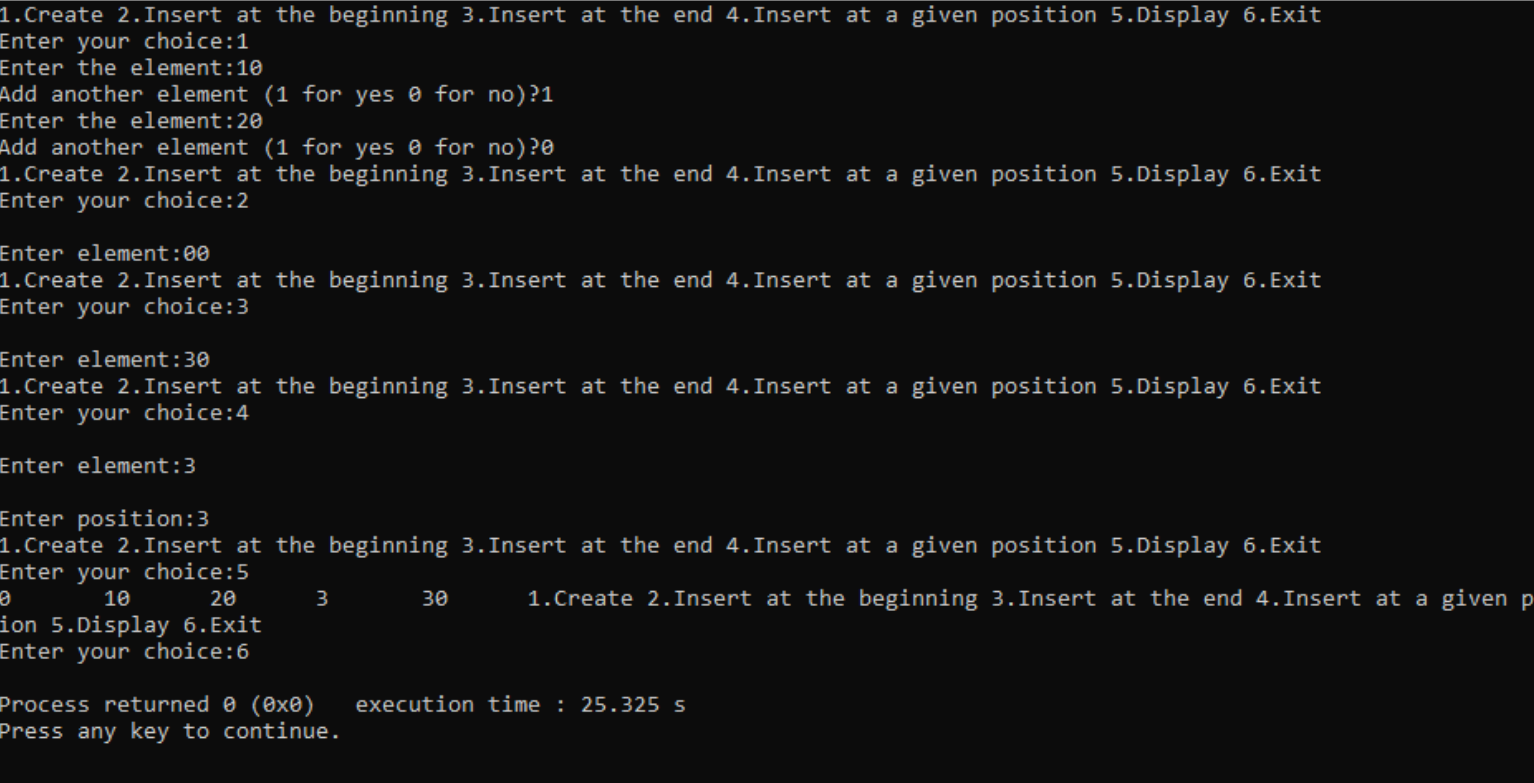
}

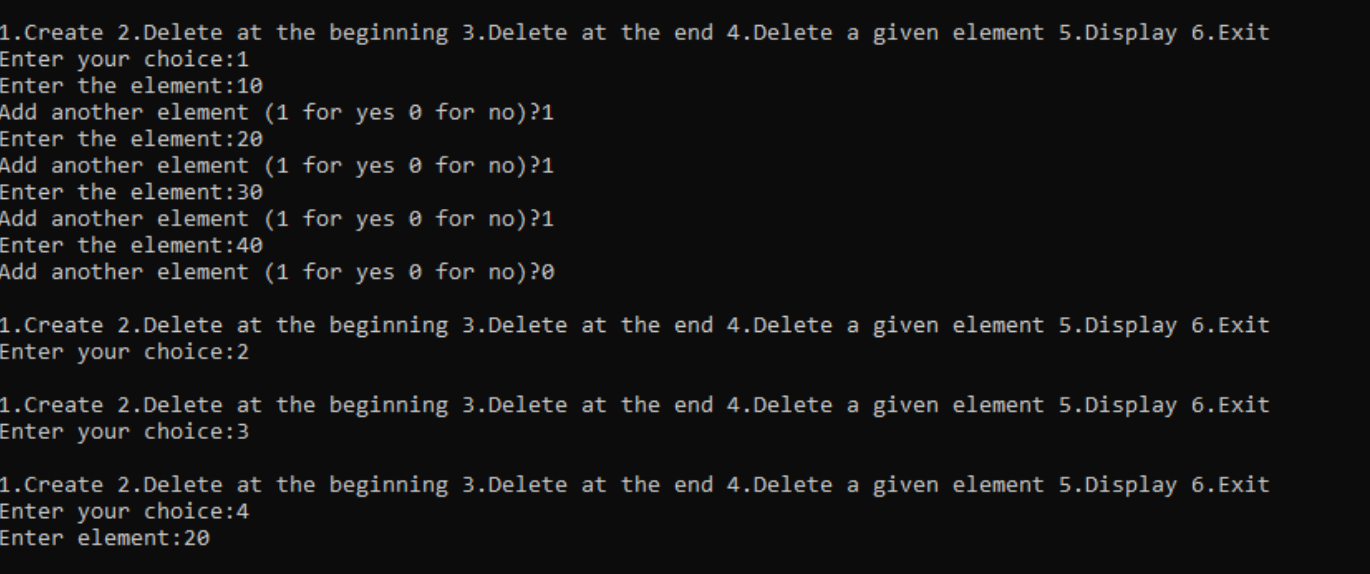
}while(choice!=9);

return 0;

}

**Sample Output :**

****

****