**Programs**

1. Write a program to implement singly linked list.

Ans:

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

struct node

{

int data;

struct node \*next;

};

struct node \*head;

void beg\_insert();

void end\_insert();

void ran\_insert();

void beg\_delete();

void end\_delete();

void ran\_delete();

void display();

void search();

void main()

{

int ch=0;

clrscr();

while(ch!=9)

{

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

printf("\n1.Begin Insert \t2.Random Insert \t3.End Insert \t4.Begin Delete");

printf("\n5.Random Delete \t6.End Delete \t7.Display \t8.Search \n9.Exit");

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

printf("\nEnter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:beg\_insert();

break;

case 2:ran\_insert();

break;

case 3:end\_insert();

break;

case 4:beg\_delete();

break;

case 5:ran\_delete();

break;

case 6:end\_delete();

break;

case 7:display();

break;

case 8:search();

break;

case 9:exit(0);

break;

default:printf("Invalid choice..");

}

}

}

void beg\_insert()

{

struct node \*ptr;

int item;

ptr=(struct node \*)malloc(sizeof(struct node \*));

if(ptr==NULL)

{

printf("\nOverflow..");

}

else

{

printf("Enter element:");

scanf("%d",&item);

ptr->data=item;

ptr->next=head;

head=ptr;

printf("Node inserted successfully..");

}

}

void ran\_insert()

{

int i,loc,item;

struct node \*ptr,\*temp;

ptr=(struct node \*)malloc(sizeof(struct node));

if(ptr==NULL)

{

printf("Overflow..");

}

else

{

printf("Enter element:");

scanf("%d",&item);

ptr->data=item;

printf("Enter the location after which you want to insert:");

scanf("%d",&loc);

temp=head;

for(i=0;i<loc;i++)

{

temp=temp->next;

if(temp==NULL)

{

printf("Insertion failed..");

}

}

ptr->next=temp->next;

temp->next=ptr;

printf("Node Inserted successfully..");

}

}

void end\_insert()

{

struct node \*ptr,\*temp;

int item;

ptr=(struct node \*)malloc(sizeof(struct node));

if(ptr==NULL)

{

printf("Overflow..");

}

else

{

printf("Enter element:");

scanf("%d",&item);

ptr->data=item;

if(head==NULL)

{

ptr->next=NULL;

head=ptr;

printf("Node inserted successfully..");

}

else

{

temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=ptr;

ptr->next=NULL;

printf("Node inserted successfully..");

}

}

}

void beg\_delete()

{

struct node \*ptr;

if(head==NULL)

{

printf("List is empty..");

}

else

{

ptr=head;

head=ptr->next;

free(ptr);

printf("Node deleted successfully..");

}

}

void ran\_delete()

{

struct node \*ptr,\*ptr1;

int loc,i;

printf("Enter location:");

scanf("%d",&loc);

ptr=head;

for(i=0;i<loc;i++)

{

ptr1=ptr;

ptr=ptr->next;

if(ptr==NULL)

{

printf("Can't delete..");

}

}

ptr1->next=ptr->next;

free(ptr);

printf("Node deleted successfully..");

}

void end\_delete()

{

struct node \*ptr,\*ptr1;

if(head==NULL)

{

printf("List is empty..");

}

else if(head->next==NULL)

{

head=NULL;

free(head);

printf("Node deleted successfully..");

}

else

{

ptr=head;

while(ptr->next!=NULL)

{

ptr1=ptr;

ptr=ptr->next;

}

ptr1->next=NULL;

free(ptr);

printf("Node deleted successfully..");

}

}

void search()

{

struct node \*ptr;

int item,i=0,flag;

ptr = head;

if(ptr == NULL)

{

printf("List is empty..");

}

else

{

printf("Enter element which you want to search:");

scanf("%d",&item);

while (ptr!=NULL)

{

if(ptr->data == item)

{

printf("Item found at location %d ",i+1);

flag=0;

}

else

{

flag=1;

}

i++;

ptr = ptr -> next;

}

if(flag==1)

{

printf("Item not found");

}

}

}

void display()

{

struct node \*ptr;

ptr=head;

if(ptr==NULL)

{

printf("List is empty..");

}

else

{

printf("Elements in linked list are:\n");

while(ptr!=NULL)

{

printf("%d\t",ptr->data);

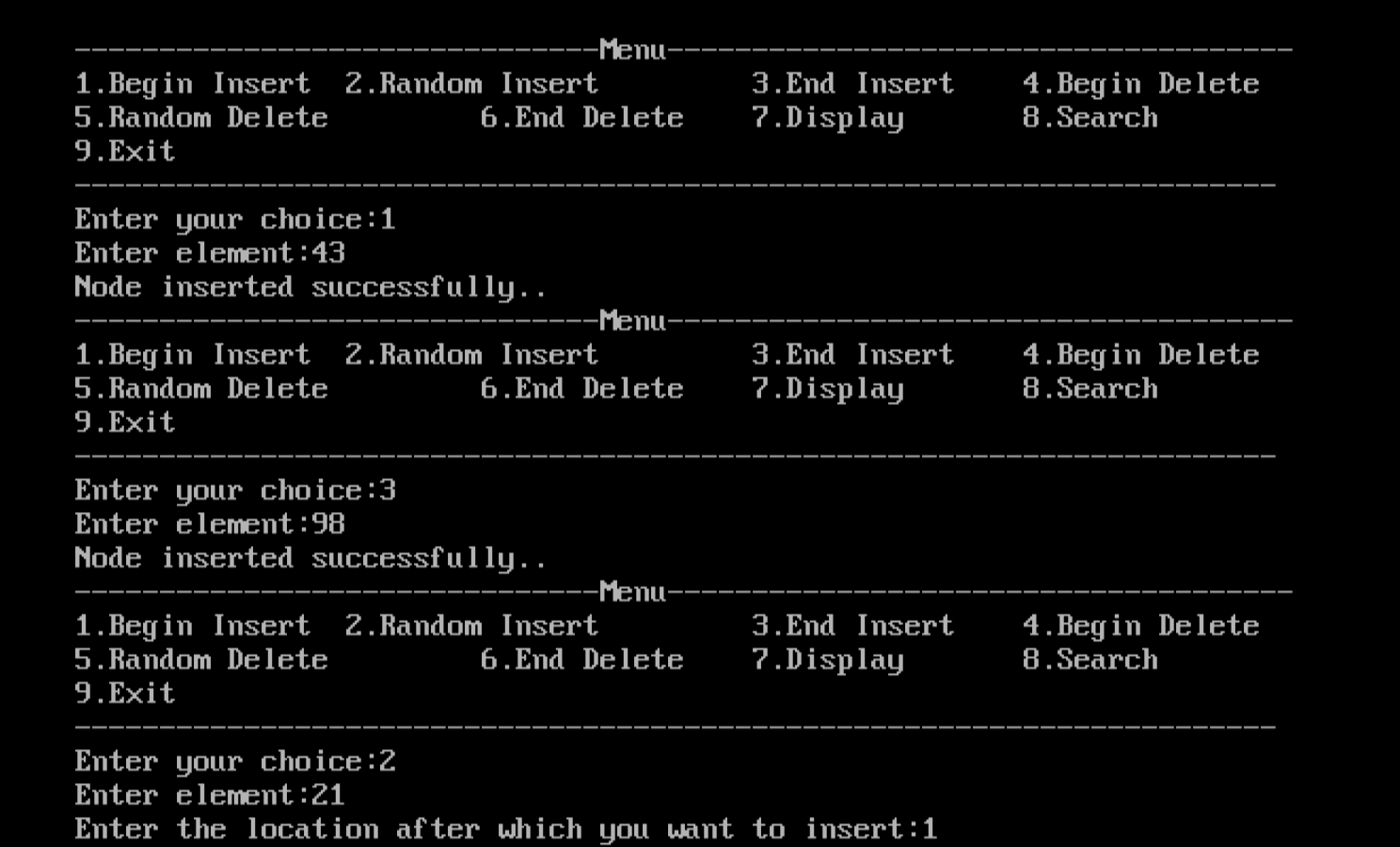
ptr=ptr->next;

}

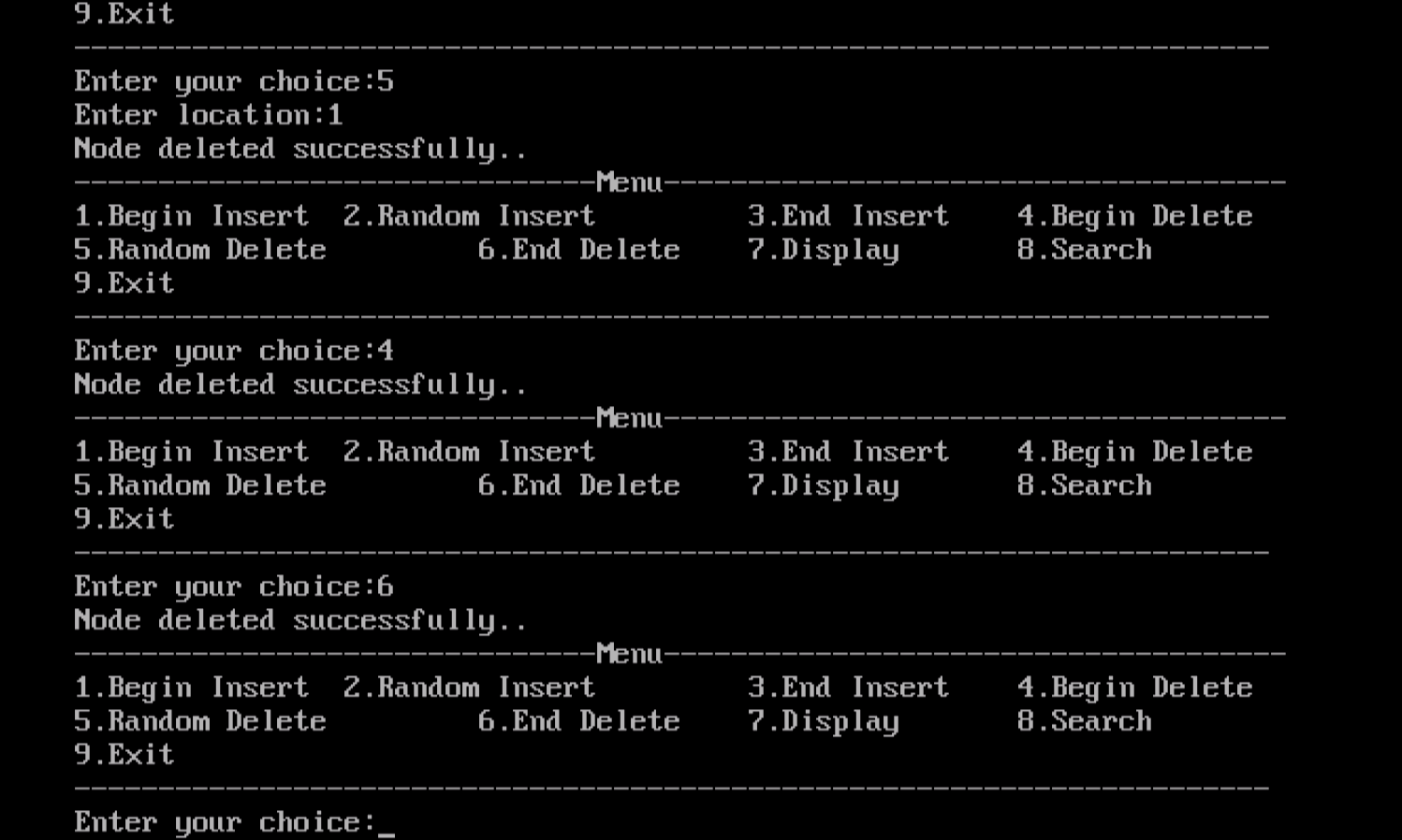
}

}

Output:







1. Write a program to implement doubly linked list.

Ans:

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

struct node

{

int data;

struct node \*next,\*prev;

};

struct node \*head;

void beg\_insert();

void end\_insert();

void ran\_insert();

void beg\_delete();

void end\_delete();

void ran\_delete();

void display();

void search();

void main()

{

int ch=0;

clrscr();

while(ch!=9)

{

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

printf("\n1.Beg\_Insert \t 2.End\_Insert \t 3.Ran\_Insert \t 4.Del\_Beg");

printf("\n5.End\_Delete \t 6.Ran\_Delete \t 7.Display \t 8.Search \t 9.Exit");

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

printf("\nEnter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:beg\_insert();

break;

case 2:end\_insert();

break;

case 3:ran\_insert();

break;

case 4:beg\_delete();

break;

case 5:end\_delete();

break;

case 6:ran\_delete();

break;

case 7:display();

break;

case 8:search();

break;

case 9:exit(0);

break;

default:printf("Invalid choice...");

}

}

}

void beg\_insert()

{

struct node \*ptr;

int item;

ptr=(struct node \*)malloc(sizeof(struct node));

if(ptr==NULL)

{

printf("Overflow..");

}

else

{

printf("Enter element:");

scanf("%d",&item);

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("Node inserted successfully..");

}

}

void end\_insert()

{

struct node \*ptr,\*temp;

int item;

ptr=(struct node \*)malloc(sizeof(struct node));

if(ptr==NULL)

{

printf("Overflow..");

}

else

{

printf("Enter element:");

scanf("%d",&item);

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("Node inserted successfully..");

}

void ran\_insert()

{

struct node \*ptr,\*temp;

int item,loc,i;

ptr=(struct node \*)malloc(sizeof(struct node));

if(ptr==NULL)

{

printf("Overflow..");

}

else

{

temp=head;

printf("Enter location:");

scanf("%d",&loc);

for(i=0;i<loc;i++)

{

temp=temp->next;

if(temp==NULL)

{

printf("Insertion failed");

}

}

printf("Enter element:");

scanf("%d",&item);

ptr->data=item;

ptr->next=temp->next;

ptr->prev=temp;

temp->next=ptr;

temp->next->prev=ptr;

printf("Node inserted successfully..");

}

}

void beg\_delete()

{

struct node \*ptr;

if(head==NULL)

{

printf("Underflow..");

}

else if(head->next==NULL)

{

head=NULL;

free(head);

printf("Node deleted successfully..");

}

else

{

ptr=head;

head=head->next;

head->prev=NULL;

free(ptr);

printf("Node deleted successfully..");

}

}

void end\_delete()

{

struct node \*ptr;

if(head==NULL)

{

printf("Underflow..");

}

else if(head->next==NULL)

{

head=NULL;

free(head);

printf("Node deleted successfully..");

}

else

{

ptr=head;

if(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->prev->next=NULL;

free(ptr);

printf("Node deleted successfully..");

}

}

void ran\_delete()

{

struct node \*ptr,\*temp;

int val;

printf("Enter element after which the node is to be deleted:");

scanf("%d",&val);

ptr=head;

while(ptr->data!=val)

ptr=ptr->next;

if(ptr->next==NULL)

{

printf("Can't delete");

}

else if(ptr->next->next==NULL)

{

ptr->next=NULL;

}

else

{

temp=ptr->next;

ptr->next=temp->next;

temp->next->prev=ptr;

free(temp);

printf("Node deleted successfully..");

}

}

void display()

{

struct node \*ptr;

printf("Elements in list are:\n");

ptr=head;

while(ptr!=NULL)

{

printf("%d\t",ptr->data);

ptr=ptr->next;

}

}

void search()

{

struct node \*ptr;

int item,i=0,flag;

ptr = head;

if(ptr == NULL)

{

printf("List is empty");

}

else

{

printf("Enter element:");

scanf("%d",&item);

while (ptr!=NULL)

{

if(ptr->data==item)

{

printf("Item found at location %d ",i+1);

flag=0;

break;

}

else

{

flag=1;

}

i++;

ptr=ptr->next;

}

if(flag==1)

{

printf("Item not found");

}

}

}

Output:



