#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*head;

//insertion at the beginning

void begininsert()

{

struct node \*ptr;

int item;

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

if(ptr=NULL)

{

printf("Overflow");

}

else

{

printf("Enter value");

scanf("%d",&item);

ptr->data=item;

ptr->next=head;

head=ptr;

printf("Node inserted");

}

//insertion at the end

void lastinsert()

{

struct node \*ptr, \*temp;

int item;

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

if(ptr==NULL)

{

printf("Overflow");

}

else

{

printf("Enter value");

scanf("%d",&item);

ptr->data=item;

if(head==NULL)

{

ptr->next=NULL;

head=ptr;

printf("Node inserted");

}

else

{

temp=head;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=ptr;

ptr->next=NULL;

printf("Node inserted");

}

}

}

//insertion at random location

void randominsert()

{

struct node \*ptr, \*temp;

int item, i, loc;

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

if(ptr==NULL)

{

printf("Overflow");

}

else

{

printf("Enter value");

scanf("%d",&item);

ptr->data=item;

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

scanf("%d",&loc);

temp=head;

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

{

temp=temp->next;

if(temp==NULL)

{

printf("Cannot insert");

return;

}

};

ptr->next=temp->next;

temp->next=ptr;

printf("Node inserted");

}

}

//deletion at the beginning

void begindelete()

{

struct node \*ptr;

if(head==NULL)

{

printf("List is empty");

}

else

{

ptr=head;

head=ptr->next;

free(ptr);

printf("Node deleted from the beginning");

}

}

//deletion at the end

void lastdelete()

{

struct node \*ptr, \*ptr1;

if(head==NULL)

{

printf("List is empty");

}

else if(head->next==NULL)

{

head=NULL;

free(head);

printf("Only node of the list is deleted");

}

else

{

ptr=head;

while(ptr->next!=NULL)

{

ptr1=ptr;

ptr=ptr->next;

}

ptr1->next=NULL;

free(ptr);

printf("Deleted node from the last");

}

}

//deletion at random location

void deleterandom()

{

struct node \*ptr, \*ptr1;

int i, loc;

printf("Enter the location of the node after which it has to be deleted");

scanf("%d", &loc);

ptr=head;

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

{

ptr1=ptr;

ptr=ptr->next;

if(ptr==NULL)

{

printf("Cannot delete");

return;

}

}

ptr1->next=ptr->next;

free(ptr);

printf("Delete node %d",loc+1);

}

//display

void display()

{

if(head==NULL)

{

printf("Linked list is empty\n");

return;

}

printf("linkedlist: ");

struct node\* ptr=head;

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

printf("\n");

}

//main

int main()

{

int choice;

while(1)

{

printf("Operation to be performed by linked list");

printf("1. Insert\n2. Begin\n3.Insert at begin\n4.Insert at end\n5.Insert at random loc\n6. Delete at begin\n7. Delete at end\n8.Delete at random loc\n9.Exit\n");

printf("\n Enter choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:begininsert();

break;

case 2:lastinsert();

break;

case 3:randominsert();

break;

case 4:begindelete();

break;

case 5:lastdelete();

break;

case 6:deleterandom();

break;

case 7:exit(0);

break;

default:printf("Invalid choice");

}

}

}