**Code for singly linked list:**

#include<stdio.h>

#include<malloc.h>

typedef struct node

{

int data;

struct node \*next;

}NODE;

typedef NODE\* NODEPTR;

NODEPTR start,newNode,tptr,shadow,tptr2;

int count;

void insert(int givenData)

{

newNode=(NODEPTR)malloc(sizeof(NODEPTR\*));

newNode->data=givenData;

newNode->next=NULL;

if(start==NULL)

{

start=newNode;

}

else

{

for(tptr=start;tptr&&tptr->data<givenData;shadow=tptr,tptr=tptr->next);

if(tptr==start)

{

newNode->next=tptr;

start=newNode;

}

else if(tptr==NULL)

{

shadow->next=newNode;

}

else

{

shadow->next=newNode;

newNode->next=tptr;

}

}

}

void delete(int givenData)

{

NODEPTR safe;

for(tptr=start;tptr&&tptr->data!=givenData;shadow=tptr,tptr=tptr->next);

safe=tptr;

if(tptr==start)//start node to deleted

{

start=tptr->next;

}

else if(tptr==NULL)

{

printf("ELEMENT NOT FOUND");

}

else //inbetween and last element to be deleted

{

shadow->next=tptr->next;

}

free(safe);

}

void display()

{

printf("\n");

for(tptr=start;tptr!=NULL;tptr=tptr->next)

{

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

count++;

}

//printf("the total no of elements :%d",count);

}

void reversedisplay(NODEPTR print)

{

if(print->next!=NULL)

{

reversedisplay(print->next);//remember this place

}

printf("%d\n",print->data);

}

void midnode()

{

NODEPTR small,large;

for(small=start,large=start;large!=NULL&&(large->next!=NULL);large=large->next->next,small=small->next);

printf("\nthe middle element is:%d\n",small->data);//it print right and middle element

//for(small=start,large=start;large!=NULL&&(large->next->next!=NULL);large=large->next->next,small=small->next);

//printf("%d",small->data);//it print left element only

}

void reversesll()

{

NODEPTR safety;

shadow=NULL;

tptr=start;

while(tptr!=NULL)

{

safety=tptr->next;

tptr->next=shadow;

shadow=tptr;

tptr=safety;

}

start=shadow;

}

void removerep(NODEPTR start)

{

NODEPTR tptr,tptr2;

tptr=start;

while(tptr->next!=NULL)

{

if(tptr->data==tptr->next->data)//we check alnerate elements only because of sorted input

{

tptr2=tptr->next->next;

free(tptr->next);//clearly note here

tptr->next=tptr2;

}

else

{

tptr=tptr->next;

}

}

}

void oddeven()

{

int val;

NODEPTR tptr,evenstart,evenend,oddstart,oddend;

evenstart=NULL;

evenend=NULL;

oddstart=NULL;

oddend=NULL;

tptr=start;

while(tptr!=NULL)

{

val=tptr->data;

if(val%2==0)

{

if(evenstart==NULL)

{

evenstart=tptr;

evenend=evenstart;

}

else

{

evenend->next=tptr;

evenend=evenend->next;

}

}

else

{

if(oddstart==NULL)

{

oddstart=tptr;

oddend=oddstart;

}

else

{

oddend->next=tptr;

oddend=oddend->next;

}

}

tptr=tptr->next;

}

if(evenstart==NULL)//for only odd terms

{

start=oddstart;

}

else if(oddstart==NULL)//for only even terms

{

evenend->next=NULL;

}

else//for combined odd and even terms

{

evenend->next=oddstart;

oddend->next=NULL;

start=evenstart;

}

}

void odev()

{

NODEPTR shadow,tptr,tptr2,save;

tptr=start;

while(tptr->next!=NULL)

{

save=tptr->next;

shadow=tptr;

for(tptr2=tptr;tptr2&&(tptr2->data%2==0);shadow=tptr2,tptr2=tptr2->next);

//above loop checks even followed next odd term

if(tptr2!=NULL&&(tptr2!=save)&&(tptr2!=tptr))

//here second condition for all cases more important

//third condition for case 5,1 here tptr equals tptr2

{

tptr->next=tptr2;

shadow->next=tptr2->next;

tptr2->next=save;

tptr=tptr->next;

}

tptr=tptr->next;

}

}

int main()

{

int num,num2;

do

{

scanf("%d",&num);

if(num==-1)

{

break;

}

else

{

insert(num);

}

}while(1);

display();

printf("enter a number to delete:\t");

scanf("%d",&num2);

delete(num2);

printf("\nAfter deletion just reversed display on list:\n");

reversedisplay(start);

midnode();

reversesll();

printf("\nAfter list itself reversed:\n");

display();

printf("\nList after duplicated deleted:\n");

removerep(start);

display();

printf("\nEven followed by odd numbers are listed:\n");

oddeven();

display();

printf("\nEven Odd combined display:\n");

odev();

display();

return 1;

}

**Output:**

