**Write a C program to remove duplicate element from sorted Linked List.**

#include <stdio.h>

#include<stdlib.h>

typedef struct Node

{

int data;

struct Node \*next;

}node;

void add(node \*list, int a)

{

node\* newnode=malloc(sizeof(node));

newnode->data=a;

node \*pos=list;

newnode->next=NULL;

if(list->next==NULL)

list->next=newnode;

else

{

while(pos->next!=NULL)

{

pos=pos->next;

}

pos->next=newnode;

}

}

void delete(node \*list)

{

node \*pos;

pos=list->next;

while(pos->next!=NULL)

{

if(pos->data==pos->next->data)

{

pos->next=pos->next->next;

}

else

{

pos=pos->next;

}

}

}

void display(node \*list)

{

node \*pos;

pos=list->next;

while(pos!=NULL)

{

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

pos=pos->next;

}

}

int main()

{

node\* list=malloc(sizeof(node));

list->next=NULL;

int a;

for(int i=0;i<10;i++)

{

scanf("%d",&a);

add(list,a);

}

delete(list);

display(list);

return 0;

}

**Write a C program to rotate a doubly linked list by N nodes.**

#include <stdio.h>

#include<stdlib.h>

typedef struct Node

{

struct Node \*prev;

char data;

struct Node \*next;

}node;

void add(node \*list,char a)

{

node\* newnode=malloc(sizeof(node));

newnode->data=a;

if(list->next==NULL)

{

newnode->prev=list;

newnode->next=NULL;

list->next=newnode;

}

else

{

node\* pos=list;

while(pos->next!=NULL)

pos=pos->next;

pos->next=newnode;

newnode->prev=pos;

newnode->next=NULL;

}

}

void display(node \*list)

{

node \*pos;

pos=list;

while(pos!=NULL)

{

printf("%c ",pos->data);

pos=pos->next;

}

}

void rotate(node\*\* list,int n)

{

node \*pos=\*list;

while(pos->next!=NULL)

{

pos=pos->next;

}

pos->next=(\*list)->next;

(\*list)->next->prev=pos;

for(int i=0;i<=n;i++)

{

\*list=(\*list)->next;

}

(\*list)->prev->next=NULL;

(\*list)->prev=NULL;

}

int main()

{

node\* list=malloc(sizeof(node));

list->next=NULL;

list->prev=NULL;

int n;

char a;

scanf("%d",&n);

for(int i=0;i<n;i++)

{

scanf(" %c",&a);

add(list,a);

}

int n1;

scanf("%d",&n1);

rotate(&list,n1);

display(list);

} #include <stdio.h>

#include<stdlib.h>

typedef struct Node

{

struct Node \*prev;

char data;

struct Node \*next;

}node;

void add(node \*list,char a)

{

node\* newnode=malloc(sizeof(node));

newnode->data=a;

if(list->next==NULL)

{

newnode->prev=list;

newnode->next=NULL;

list->next=newnode;

}

else

{

node\* pos=list;

while(pos->next!=NULL)

pos=pos->next;

pos->next=newnode;

newnode->prev=pos;

newnode->next=NULL;

}

}

void display(node \*list)

{

node \*pos;

pos=list;

while(pos!=NULL)

{

printf("%c ",pos->data);

pos=pos->next;

}

}

void rotate(node\*\* list,int n)

{

node \*pos=\*list;

while(pos->next!=NULL)

{

pos=pos->next;

}

pos->next=(\*list)->next;

(\*list)->next->prev=pos;

for(int i=0;i<=n;i++)

{

\*list=(\*list)->next;

}

(\*list)->prev->next=NULL;

(\*list)->prev=NULL;

}

int main()

{

node\* list=malloc(sizeof(node));

list->next=NULL;

list->prev=NULL;

int n;

char a;

scanf("%d",&n);

for(int i=0;i<n;i++)

{

scanf(" %c",&a);

add(list,a);

}

int n1;

scanf("%d",&n1);

rotate(&list,n1);

display(list);

}

**Write a C program to sort the elements of a queue in ascending order.**

#include <stdio.h>

#include<stdlib.h>

typedef struct node

{

int data;

struct node\* next;

}node;

node \*front=NULL;

node \*rear=NULL;

void add(int ele)

{

node \*newnode=malloc(sizeof(node));

newnode->data=ele;

newnode->next=NULL;

if(rear==NULL)

front=rear=newnode;

else

{

rear->next=newnode;

rear=newnode;

}

}

void display()

{

node \*tmp=front;

while(tmp!=NULL)

{

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

tmp=tmp->next;

}

}

void swap(node \*p1,node \*p2)

{

int tmp = p2->data;

p2->data = p1->data;

p1->data = tmp;

}

void sort(int n)

{

node\* p1;

for(int i=0;i<n;i++)

{

int swapped=0;

p1=front;

for(int j=0;j<n-i-1;j++)

{

if(p1->data>p1->next->data)

{

swap(p1,p1->next);

swapped=1;

}

p1=p1->next;

}

if(swapped=0)

break;

}

}

int main()

{

int n,a;

scanf("%d",&n);

for(int i=0;i<n;i++)

{

scanf("%d",&a);

add(a);

}

sort(n);

display();

return 0;

}

**List all queue function operations available for manipulation of data elements in c**

Enqueue

Dequeue

Isfull

Isempty

Peek

Traverse

display

**Reverse the given string using stack**

#include <stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct node

{

int data;

struct node\* next;

}node;

void add(node \*\*list,char ch)

{

node\* newnode=malloc(sizeof(node));

newnode->data=ch;

if(\*list==NULL)

{

\*list=newnode;

newnode->next=NULL;

}

else

{

newnode->next=\*list;

\*list=newnode;

}

}

void display(node \*list)

{

node \*tmp=list;

while(tmp!=NULL)

{

printf("%c",tmp->data);

tmp=tmp->next;

}

}

int main()

{

char ch[100];

scanf("%s",ch);

node\* list=NULL;

for(int i=0;i<strlen(ch);i++)

{

add(&list,ch[i]);

}

display(list);

}

**Insert value in sorted way in a sorted doubly linked list. Given a sorted doubly linked list and a value to insert, write a function to insert the value in sorted way.**

#include <stdio.h>

#include<stdlib.h>

typedef struct Node

{

struct Node \*prev;

int data;

struct Node \*next;

}node;

void add(node \*list,int a)

{

node\* newnode=malloc(sizeof(node));

newnode->data=a;

if(list->next==NULL)

{

newnode->prev=list;

newnode->next=NULL;

list->next=newnode;

}

else

{

node\* pos=list;

while(pos->next!=NULL)

pos=pos->next;

pos->next=newnode;

newnode->prev=pos;

newnode->next=NULL;

}

}

void add1(node \*list,int a)

{

node\* newnode=malloc(sizeof(node));

newnode->data=a;

node\* pos=list->next;

while(a>(pos->data))

{

pos=pos->next;

}

node\* tmp=pos->prev;

newnode->prev=tmp;

newnode->next=tmp->next;

tmp->next=newnode;

}

void display(node \*list)

{

node \*pos;

pos=list->next;

while(pos!=NULL)

{

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

pos=pos->next;

}

}

int main()

{

node\* list=malloc(sizeof(node));

list->next=NULL;

list->prev=NULL;

int n;

int a;

scanf("%d",&n);

for(int i=0;i<n;i++)

{

scanf("%d",&a);

add(list,a);

}

int n1;

scanf("%d",&n1);

add1(list,n1);

display(list);

}

**Write a C program to insert/delete and count the number of elements in a queue.**

#include <stdio.h>

#include<stdlib.h>

typedef struct node

{

int data;

struct node\* next;

}node;

node \*front=NULL;

node \*rear=NULL;

void add(int ele)

{

node \*newnode=malloc(sizeof(node));

newnode->data=ele;

newnode->next=NULL;

if(rear==NULL)

front=rear=newnode;

else

{

rear->next=newnode;

rear=newnode;

}

}

void delete()

{

if(rear==NULL)

printf("no element");

else

{

node\* tmp=front;

if(front==rear)

{

front=rear=NULL;

}

else

{

front=front->next;

}

free(tmp);

}

}

void display()

{

node \*tmp=front;

printf("Queue elements are:");

while(tmp!=NULL)

{

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

tmp=tmp->next;

}

printf("\n");

}

void count()

{

node \*tmp=front;

int count=0;

while(tmp!=NULL)

{

count++;

tmp=tmp->next;

}

printf("Number of elements in the Queue:%d\n",count);

}

int main()

{

int n,a;

printf("enter the choice:");

scanf("%d",&n);

do{

switch(n)

{

case 1:

printf("enter the element:");

scanf("%d",&a);

add(a);

break;

case 2:

delete();

break;

case 3:

display();

break;

case 4:

count();

break;

}

printf("enter the choice:");

scanf("%d", &n);

}while(n!=0);

return 0;

}

**Write a C program to Find whether an array is a subset of another array.**

#include <stdio.h>

int main()

{

int n1,n2;

scanf("%d",&n1);

int a[n1];

for(int i=0;i<n1;i++)

scanf("%d",&a[i]);

scanf("%d",&n2);

int b[n2];

for(int i=0;i<n2;i++)

scanf("%d",&b[i]);

int count=0;

for(int i=0;i<n2;i++)

{

for(int j=0;j<n1;j++)

{

if(b[i]==a[j])

{

count++;

break;

}

}

}

if(count==n2)

printf("arr2[] is a subset of arr1[]");

else

printf("arr2[] is not a subset of arr1[]");

return 0;

}