Write a program to insert and delete and element at element at element at the n^{th} and k^{th} pointer in a linked list where n and k are taken from the users.

```
#include <stdio.h>
#include <stdlib.h>
struct node {
int data;
struct Node next;
};
struct Node * head;
void Insert (int data, int n) {
Node *temp = new node ();
temp -> data = data;
temp -> next = null;
if (n==1){
  temp -> next = head;
  head = temp;
  return;
}
void delete (int k){
struct Node * temp = head;
if (k==1){
  head = temp -> next;
free (temp);
  return;
}
Node * temp = head;
for (int i=0; i<n-2; i++){
 temp = temp -> next;
}
temp-> next = temp -> next;
temp -> next = temp;
}
Void print();
for (int i=0; i<k-2; i++)
temp = temp -> next;
free (temp);
}
```

```
int main() {
int n, x, k;
head = NULL;
printf(" Enter the position for interesting : ");
scanf ("%d", &n);
scanf ("%d", &x);
Insert (x, n);
printf (" Enter the position to delete");
scanf ("%d", &k);
Delete (k);
Print (x);
return;
}
```

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node next;
}
void print list (struct node* head)
{
printf ("%d -> ", (ptr->data));
ptr = ptr -> next;
printf ("NULL \n");
}
void push (struct node * head, int data)
{
struct node* new = (struct node) malloc (sizeof (struct node));
new -> data = data;
new -> next = * head;
* head = new;
}
struct node * merge (struct node * a, struct node * b)
{
struct node fake;
struct node * tail = fake;
fake.next = null;
while (1) {
if (a == null)
{
```

```
tail -> next = b;
break;
}
else if (b==null)
{
  tail -> next = a;
  break;
}
else
{
  tail -> next = a;
  tail = a;
  a = a -> next;
  tail -> next = b;
  }
}
return fake next;
}
void main()
{
int keys[] = {1,2,3,4,5,6,7}
int n = size of (keys) / size of key[0]
struct node * a = null; *b = Null;
for (int i = n-1, i>0 ; i=i-a)
   Push (fa, keys[i]);
for (int i<n-2; i>=0 ; i=i-2)
   push (fb ; key[i]);
struct node * head = merge (a,b))
 print List (head);
}
```

⇔

```
#include <stdio.h>
int top =-1;
int x;
char stack[100];
void push(int x);
char pop()
int main()
{
int i, n, a, t, k, f, sum=0, count = 1;
printf("Enter the number of elements in the stack");
scanf("%d", &n);
for (i=0 ; i<n ; i++) {
printf ("Enter next element");
scanf ("%d", &a );
push(a);
}
printf("Enter the sum to be checked");
scanf ("%d", &k);
for (i=0 ; i<n; i++)
{
t = pop();
sum = t;
count += 1;
if (sum == k) {
for (int j=0; j<count; j++)
printf ("%d", stack[j]);
f--;
break;
}
push (t);
}
if (f != 1)
   printf ("The elements in the stack don't add up to the sum");
}
void push(int x)
{
if (top == 99)
{  printf ("\n stack is full !!\n");
   return;
}
```

```
top=top+1;
stack[top]=x;
}
char pop()
{
if (stack[top]=x)
}
charpop()
{
if (stack[top]==-1)
{printf("\n stack empty")
return 0;
}
x= stack[top];
top--;
}
```

⇒
```
#include <stdio.h>
# define SIZE 10
void insert (int);
void delete();
int queue[10], f=-1, r=-1;
void main(){
int value, choice;
while(1){
printf("\n\n\t MENU * \n");
printf("1. Insertion\n 2. Deletion\n 3.Reverse\n 4.Alternate");
printf("\n enter your choice");
scanf("%d", &choice);
if(choice==1){
Printf("Enter the value to be insert:");
scanf("%d", &value);
insert(value); break;}
```

```
else if (choice==2)
{ delete();
  break;
}
else if (choice==3)
{ printf("The Reversed queue is");
  for (int i=size; i>=0 ; i--)
{ if (queue[i]==0)
  continue;
  printf("%d", queue[i]);
}
  break;
}
else if (choice==4)
{
  printf("Alternate elements of queue");
  for (int i=0; i<size ; i+=2)
  { if (queue[i]==0)
    continue;
    printf("%d", queue[i]);
  }
  break;
  }
else if (choice==5)
    exit(0);
else {
    printf("wrong selection");
}
}
void insert(int value){
if ((f==0 && r==size-1) || f==r+1)
    printf("\n Queue is full");
```

```
else {
  if (F==-1)
  F=0;
  r = (r+1) % SIZE;
  queue[r] = value;
  printf("\n Insertion Success!!");
}
void delete(){
  if (F==-1)
  printf("\n Queue is Empty");
else {
  printf("\n Deleted : %d", queue[F]);
  F =(F+1) % SIZE;
  if (F==r)
  F = r= -1;
  }
```

⇒

→i, How array is different from the linked list?

The major difference between Array and linked list regards to their structure. Arrays are index based data structure where each element associated with an index. on the other hand, linked list relaif on reference to the previous and nax element.

→ii,

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
  int data;
   struct node * next;
};
```

```
void push (struct node ** head_ref, int new_data)
{
struct node* new_node = (struct node*) malloc (sizeof(struct node));
new_node -> data = new_data;
new_node -> next = (*head_ref);
(* head_ref ) = new_node;
}
void printList (struct node* head)
{
 struct node * temp = head;
 while (temp != NULL)
 {
  printf ("%d", temp-> data);
  temp = temp->next;
 }
 print ("\n");
}
```