Write a program to insert and delete an element at the n^{th} and k^{th} position in a linked list where n and k is taken from user.

```
#include <Stdio.h>
#include <malloc.h>
#include <Stdlib.h>.
Struct node{
        int value;
        Struct node * next ;

};
void insert();
void display();
void delete();
int Count();
type def Struct node DATA-NODE;
DATA-NODE* head-node* First-node ,* temp-node=0,* Previous-node
next-nod;
int data;
Int main()
{
  int choice=0
  while (choice<6){
  printf ("\n Choices\n");
  printf ("1:Insert\n");
```

```
printf(" 2: Delete \n");
printf(" 3; Display \n");
printf(" 4: Count linked list \n");
printf(" others: Exit()\n");
printf(" Enter your choice ");
scanf(" %d", &choice);
switch (choice){
    case 1: insert();
            break;
    case 2: delete();
            break;
    case 3: display();
            break;
    case 4: count();
            break;
    default:
            break;
    }
}
return 0;
}
void insert() {
        printf(" In Enter Element for Insert linked list:
                \n");
```

```
scanf ("%d", &data);
temp-node = (DATA-NODE*) malloc (sizeof (DATA-NODE));
temp-node -> value = data;
if (first-node == 0)
{
    first-node = temp-node;
}
else
    {
      head-node -> next = temp-node;
    }
temP-node -> next = 0;
 head-node = temp-node
 fflush (stdin);
}
void delet () {
 int count value, position, n=0
   count value = count ();
   temp-node = first-node;
   Printf ("In display linked list: \n");
   Printf ("Enter position to delete: \n");
   Scanf ("%d", &position);
```

```
if (position >>0 && pos <= count value){
{ if (position == 1)
    {
      temp_node = temp_node -> next ;
      four_node = temp_node',
      printf (" Element deleted \n");
{
else
{
while (temp_ node != 0)
{
  if (i == (position-1)
    {
      prev_node -> next = temp_node -> next ;
      if (n == (count value -1));
      {
        head_node = previous_node',
      }
      printf (" Deleted ");
      break;
    }.
```

```
else
  {
    i++;
      Previous_node = temp.node
    temp-node =temp_nod -> next;
      }
    }
  }
  else
  {
    printf("Invalid \n\n");
    }
  }
void display ()
  {
    int count=0;
    temp_node = first_node;
    first
    printf("\n Display : \n");
    while (temp_node! =0)
    {
      printf ("%d", temp_node -> value);
      count ++;
      temp -node = temp-node -> next;
    }
```

printf ("\n no. of items : %d \n", count);

}

int count()

{

int count = 0;

temp_node = first->node;

while (temp_node != 0){

count++;

temp_node = temp_node -> next;

}

printf ("\n no. of items %d \n", count);

return count;

}

(or) Construct a new linked list by merging alternates nodes of two lists for example in list 1 we have {1,2,3} and in the list 2 we have {4,5,6} in the new list we should have {1, 4, 2, 5, 3, 6}.

Sol)

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
struct node
{
```

```
int data;
Struct node * next;
};
void move node (Struct node ** x ; struct node **y);
Struct node * Sorted merge (struct node *a ,structnode*b);
{
   Struct node dall;
   Struct node * tail = & dall;
   dall.next = Null;
   while(1)
   { if (a==Null)
     {
        tail -> next =b;
        break;
     }
    else if (b==Null)
    {
       tail -> next=a;
       break;
    }
    if (a->Data <=b -> data)
    {
       move node (&(tail) -> next), &a);
```

```
}
else
{
    move node = (&(tail)->next, &b);

    y
    tail = tail->next,

    y
    return (dall.next)
void move node *(struct node **x, struct node** y)

{
    struct node* newnode = *y;
    assort (new node != NULL);
    * if = new node -> next;
    new node -> next = * x;
    * x == new node;
y
void push (struct node :** head_ref, int new_data)
{
struct node * new_node = (struct node*) malloc (size of
                                    (struct node));
    new -> node -> data = new -> data;
    new - node -> next = (* head -ref);
```

```
{
 while (node != null)
  {
    printf ("%d ", node ->data);
     node = node ->next ;
  }
 }
int main()
{
Struct node * res = null ;
 Struct node * a = null;
 Struct node * b = null;
 Push (&a,1);
 Push (&a,2);
 Push (&a, 3);
 Push (&a,4);
 Push (&b , 5)
Push (&b,6)
 res =Sorted merge (a,b).
 printf (" merge linked list is: \n");
  print f list (res);
  return 0;
}
```

(3) Find all the elements in the stack whose sum is equal to k.

```
#include <stdio.h>
int top = -1;
int a;
char stack[50];
void push(int a);
char pop();
int main()
{
 int n, i, x, t, k, f, sum=0, count=1;
 printf("Enter the number of elements");
 scanf("%d", &i);
 for(n=0 ; n<i; n++)
 {
   printf("Enter next element");
   scanf("%d", &a);
   Push(a);
 }
printf("Enter the sum to be checked");
scanf("%d", &k);
for(n=0; n<i; n++)
```

```
{
  t = POP()
   Sum+= t;
   count+ = 1;
   if (sum = = k)
   {
       for (int  j =0;  j <count ;j++)
        printf(" %d ", stack [j]);
   f =1;
   break;
   }
   Push (t);
   }
printf (" The element in the stack dont add up ");
}

void Push (int a)
{
    if (top = =qa)
      {
   printf(" In Stack is Full !!!\n");
    return;
}
return
top= top+1
  Stack [top ] =x;
}
```

```
char POP()
{
  if (stack [TOP] == -1)
    {
      printf ("\n Stack is EMPTY");
      return 0;
    }
  x = Stack [TOP];
  TOP = top-1)
  return x;
}
```

4) Write a program to print the elements in a queue
   i) reverse order
   ii) insertion order

```
# 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 * * * MENU * * * \n");
Printf("1. Insertion 2. deletion \n 3. Reverse \n 4. Alternate
                                        \n 5. Exit");
Printf ("\n Enter your choice");
scanf ("%d", & Choice);
 Switch (Choice)
{
  case 1:
 Printf (" Enter the value to be insert ");
scanf ("%d", & value);
 Insert (value);
 break;
 case 2: delete ();

  break;
 case 3:
 Printf ("The reversed queue is :");
for (int i=size ; i>= 0; i++)
{
  if (queue [i] =0=0
  continue;
  Printf ("%d", queue [i]);
```

```
{
    }
of  break;
    case 4:
    Print 4 C" Alternate elements of the queue :"J;
    for (int i=0 ; i< size ; i+=2)
    {
        if (queue [i] == 0)
         continue ;
         Print f (" %d ", queue [i] J;
        }
        break ;
        case 5: exit (0);
        default : print f (" \n wrong selection "J;
        }
    }
}
void insert (int( value )
{
    if ((f == 0 && r == size-1) || f == r+1)
        Print (" In Queue is full "J;
    else
```

```
{
  if (f==-1)
     f=0;
     r=(r+1)%.size;
     queue[r]=value;
     Printf("\n Insertion done");
   }
}
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;
  }
}
```

(5) (i) How array is different from the linked list.

(ii) Write a program to add the first element of one list to another list for example we have {1,2,3} in list1 and {4,5,6} in list 2 we have to get {4,1,2,5} as output for list1 and {5,6} for list2.

Sol) (i) —) difference in their structure

—) arrays are index based.

—) linked list relies on reference to the previous and next element.

(ii) #Include (stdio.h)

#Include (Stdlib.h)

Struct node.

{

int data;

Struct node * next;

}

void push (Struct node * hed-ref, int new-data)

{

Struct node ** new-node = (Struct-node *) malloc

(Sizeof(structnode));

new-node → data = new-data;

new-node → next = (*head-ref);

```
(*head_ref) = new_node
}
void print_list (struct node *head)
{
  struct node * temp = head;

    while (temp != Null)
{
    printf ("%d", temp->data);
    temp = temp->next;
}
  printf ("\n");
}
```