4 Write a program to insert and delete an element at the nth and kth position in a linked list where n and k are taken from user

program:

```
#include<stdio.h>
#include<stdlib.h>
{
struct node *next;
};
struct node *cur, *temp;
void input(struct node*)
void input(struct node*)
void main(void)
{
 struct node* s;
 int ch;
 s=Null;
 do
 {
 printf("1.Enter a number of insertion:");
 printf("2.Delete");
 printf("3.Exit.");
 printf("Enter your choice");
 scanf("%d",&ch);
    switch(ch)
 {
 case 1: input(s);
         break;
 case 2: delete(s);
         break;
```

```
    }
  While(n!=3)
}
void input(struct node* z)
 {
 int p, c=1
 curr=z;
 printf("Enter the number to insert");
 scanf("%d",&Pos);
 while(curr -> next != Null)
 {
 c++;
 if(c==Pos)
 {
temp=(struct node*)malloc(size of(struct node));
printf("Enter the elements:");
scanf("%d",&temp->ch);
    temp -> next = curr -> next;
     curr -> next = temp;
     break;
     }
   }
  }
void delete(struct node*z)
 {
  int pos, c=1;
  curr = z;
  printf("Enter the element to delete");
  scanf("%d",&Pos)
  while(curr -> next != Null).
  {
```

```
    c++;
    if(c==pos)
    {
   temp=curr->next;
   curr->next = curr->next->next;
   free(temp)
   }
  curr=curr->next;
  }
void merge (struct node *p, struct node *q)
{
struct node*p_curr=p, *q_curr=*q;
struct node*p_next,*q_next;
while(p_curr=Null && q_curr!=Null)
{
p_next=p_curr->next;
q_next=q_curr->next;
q_curr->next=p_next;
p_curr->next=q_curr;
p_curr=p_next;
q_curr=q_next;
}
*q=q_curr
}
int main()
{
Struct node*p=Null,*q=Null;
push(&p,1);
push(&p,2);
```

```
push(&P,3);
printf(" First linked list :");
print list(R);
push (&q ,1)
Push (&q, 6)
push (&q ,7)
printf("second linked list :\n");
print list(q);
merge(P,&q);
Printf("Modifide first linked list=");
print list(P);
printf("Modifide second linked list=");
print list (q);
return 0;
}
```

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

Program:-

```
#include<stdio.h>
#include<stlib.h>
#include< assert.h>
struct node
{
int data;
struct node * next;
};
void move node(struct node **a; struct node**b,
structnode* sorted merge(struct node *x, strude node*
{
struct node dummy;
struct node * tail= & dummy;
dummy. next= Null;
while(1)
{
if (a=Null)
{
*b= newnode ->next;
newnode -> next =*a;
*a =newnode;
}
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);
(* head_ref) = new node;
}
void print list( struct node * node)
{
while(node != Null)
{
printf("%d", node ->data);
  node = node ->next;
  }
}
tail -> next = y;
  break;
 }
else if (y == Null)
{
tail -> next = x;
  break;
}
if (x -> data <= y -> data)
{
move node (&(tail) -> next), &x);
}
else
{
```

```
    move node (&(tail) -> next, &b);
}
tail = tail -> next;
}
return(dummy next);
}
void. move node *(struct node ** a, struct node ** b)
{
struct node* new node = *b;
assert(new node != Null);
int main( )
{
struct node * res=Null;
struct node * x = Null;
struct node * y = Null;
push(&x, 1);
push(&x, 2);
push(&x, 3);
push(&x, 4);
push(&x, 5);
push(&x, 6);
res = sorted merge (x, y);
printf ("Merge linked list is :");
print list (res);
return 0;
}
```

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

Program:-

```
#include<stdio.h>
int sa[100], topa=-1, sb[100], topb=-1;
int sa empty()
{
if (topa == -1)
    return 1;
else
    return 0;
}
int sa top()
{
 return sa[topa];
}
int sa pop()
{
  topa--;
}
int sa push(int x)
{
  sa[++topa]=x;
}
int sb empty()
{
  if(topb==-1)
   return 1;
```

```
  else
   return 0;
}
int sb top()
{
  return sb[topb];
 }
 int  sb pop()
 {
  topb--;
 }
 int sb push(int x)
 {
 sb[++topb]=x;
 }
 int sum(int k)
 {
    int x;
    while(sa empty()!=1)
    {
     x=sa top();
      s1 pop();
    while(s1 empty()!=1)
    {
     if(x+sa top()=k)
     {
     printf("%d%d",x,sa top());
     }
    sb push(sa top());
    sa pop();
```

```
}
while(sb.empty()!=1)
{
sa.push(sb.top());
 sb.pop();
 }
}
int main()
{
 int  n,i,e,k;
  printf("Enter the number of elements ");
  scanf("%d",&n);
  for(i=0;i<n;i++)
  {
  scanf("%d",&e);
    sa.push(e);
  }
  printf("Enter the value of constant sum:");
  scanf("%d",&k);
  printf("The combinations whose sum is equal to
          k is:\n");
  sum(k);
  }
```

4) Write a program to print the elements in a queu

// in reverse order
// in alternative order

```
#include<stdio.h>
#include"stack.h"
#include "QQ.h"
int main()
{
int n,a[20],i,j=0;
struct stack s;
initstack(&s);
printf("Enter number:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter values");
scanf("%d",&a[i]);
}
for(i=0;i<n;i++)
{
insert(a[i]);
}
while(j!=n)
{
push(&s,del());
j++;
}
print("Reverse is");
while(stop!=-1)
{
```

```
printf("%d", pop(&s));
}
printf("\n");
return 0;
}
```

```
#include<stdio.h>
#include<stdlib.h>
struct node{
  int data;
struct node * next;
}
void print nodes(struct node * head)
{
int count=0;
while (head!=Null){
if (count %2==0)
{
printf("%d",head->data);
}
count++;
head=head->next;
}
}
void push(struct Node ** head_ref, int new_data)
{
struct node* new_node = (struct node *)
                        malloc(sizeof(struct node));
newnode->data = new_data;
newnode -> next = (*head_ref);
(*head_ref) = new_node;
```

```
}
int main()
{
struct node* head=Null
    push(&head,1);
    push(&head,3);
    push(&head,5);
    push(&head,9);
    push(&head,10);
    print node(head);
    return 0;
    }
```

9. How array is different from linkedlist

Array	Linked list
It is a consistent set of a fixed number of data items	It is an ordered set comprising a variable number of data items
Element location is allocated during compile time	Element position is assigned during run time

10.
```
#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(size of(struct node());
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");
}
```