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

Sol
```
#include <stdio.h>
#include <stdlib.h>
Struct node
{
Struct node *next;
};
Struct node *curr, *temp;
Void input(struct node*)
Void delete(stud node*)
Void main(void)
{
Struct node *s;
int h;
s=NULL;
do
{
Printf("1. Enter the element to insert; \n");
Printf("2. Delete \n");
Printf("3. Exit \n");
Printf("Enter the choice; ");
Scanf("%d", &h);
        switch(h)
        {
        case 1; input(s);
                break;
        case 2: delete(s);
```

```
        break;
    }while (ch != 3)
}
void input(struct node *z)
{
 int pos, c=1;
 curr = z;
 printf(" Enter the element to be inserted: ");
 scanf("%d", &pos);
        while( curr -> next != NULL)
        {
         c++;
         if (c == pos)
         {
          temp= (struct node*) malloc (size of (struct node));
          printf (" Enter the number: ");
          scanf (" %d", &temp -> n);
          temp -> next = curr -> next;
          curr -> next = temp;
          break;
         }
        }
}
void delete ( struct node *z)
{
  int pos, c=1;
  curr = z;
 printf (" Enter the element to be deleted: ");
  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:\n");
 Print List (R);
 Push (&q, 4);
 Push(&q, 5);
 Push(&q, 6);
 Printf("second Linked List:\n");
 PrintList(q);
 merge(P, &q);
 Printf(" Modified First Linked List = \n ");
 PrintList(P);
 Printf("Modified second Linked List = \n");
 Print List(q);
 return 0;
}
```

2. Construct a new linked list by merging alternative nodes of two Lists for example in List 1 we have {1,2,3} and in list2 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 MoveNode(struct node **x; struct node **y);
struct node * sortedMerge(struct node *a, struct node *b)
{
  struct node dummy;
  Struct node *tail = &dummy;
  dummy.next= NULL;
  while(1)
  {
    if(a== NULL)
    {
      tail→next=b;
      break;
    }
    else if(b==NULL)
    { tail→ next=a;
      break;
    }
```

```
if (a->data <= b->data)
{
  movenode {&(tail->next), &a);
           }

  else
   {
     Movenode (&(tail->next), &b);
   }
     tail= tail->next;
   }
   return (dummy.next);
}
void Movenode( struct node **x, struct node **y)
 {
    struct node* newnode = *y;
     assert(newnode != NULL);
     *y = newnode -> next;
    newnode -> next = *x;
     *x = new node;
  }
void Push(struct node **head_ref, int new_data)
  {
    struct nod* new_node=(struct node*)malloc(size of
                                    (struct node));
    new_node -> data= new_data;
    new_node -> next= (*head_ref);
     (*head_ref)= new_node;
  }
```

```
Void PointList (struct node * node)
{
    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(&b, 2);
    Push(&a, 3);
    Push(&b, 4);
    push(&b, 5);
    Push(&b, 6);
    res = sorted Merge(a, b);
    Printf("Merge Linked List is: \n");
    Print list (res);
    return 0;
}
```

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

Sol
```
#include <stdio.h>
int s1[10], top = -1, s2[10], top2 = -1;
int slempty()
{
  if (top == -1)
      return 1;
  else
      return 0;
}
int s1top()
{
  return s1[top1];
}
int s1pop()
{
  return top1--;
}
int s1push(int x)
{
  s1[++top1] = x;
}
int s2empty()
{
  if (top2 == -1)
      return 1;
  else
      return 0;
}
```

```
int s2top()
{
   return s2[top2];
}
int s2pop()
{
   top2--;
}
int s2push(int x)
{
   s2[++top2]=x;
}
int sum(int k)
{
   int x;
   while(s1empty()!=1)
   {
      x=s1top();
      s1pop();
      while(s1empty()!=1)
      {
         if(x+s1top()==k)
         {
            printf("(%d,%d)\n",x,s1top());
         }
         s2push(s1top());
         s1pop();
      }
      while(s2empty()!=1)
      {
         s1push(s2top());
         s2pop();
      }
   }
}
```

```
int main()
{
    int n,i,e,k;
    Printf("enter the no of elements of stack:\n");
    scanf("%d", &n);
    for(i=0; i<n; i++)
    {
        scanf("%d", &e);
        sl push(e);
    }
    Printf("enter the value of constant sum:\n");
    scanf("%d", &k);
    printf("The combinations whose sum is equal to k is:\n");
    Sum(K);
}
```

Q4 Write a program to print the elements in a queue.

i) in reverse order.

Sol
```
#include<stdio.h>
#include<stdlib.h>
Struct node
{  int a;
   Struct node * next;
};
void generate (struct node**);
void display (Struct node*);
void stack_reverse (struct node **, Struct node**);
Void delete (Struct node **);
int main()
{
   struct node *head = NULL;
   generate (&head)
   printf ("\n The sequence of contents in stack: \n");
   display(head);
   Printf("\n Inreasing the contents of the stack: \n");
   if (head != NULL)
   { stack_reverse (& head, &(head → next));
   }
  Printf ("\n the contents in stack after reversal \n");
   display(head);
   display (&head);
   return 0;
}
void stack_reverse (Struct node **head, struct node **head_next)
{
```

```
struct node *temp;
if(*head_next != NULL)
{
 temp = (*head_next)->next;
  (*head_next)->next = (*head);
   *head= *head_next;
   *head_next = temp;
  stack_reverse(head, head_next);
   }
}
void display (struct node *head)
{ if(head != NULL)
   {
     printf("%d", head->a);
     display(head->next);
    }
 }
void generate (struct node **head)
{
   int num, i;
  struct node *temp;
  Printf(" Enter length of list:");
  scanf("%d", &num);
  for(i=num; i>0; i--)
  {
   temp=(struct node*) malloc(sizeof(struct node));
   temp->a= i;
```

```
    if (*head == NULL)
    {  *head = temp;
       (*head)->next = NULL;
    }
    else
    {
      temp->next = *head;
      *head = temp;
     }
    }
}
void delete( struct node **head)
 {
    struct node *temp;
    while (*head != NULL)
     {
       temp = *head;
        *head = (*head)->next;
       free(temp);
     }
  }
```

Q4 Write a program to print the elements in a queue
ii) in alternative order

Sol
```
#include <stdio.h>
#include <stdlib.h>
struct node{
    int data;
    struct Node* next;
};
Void printnodes(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));
  new_node->data = new_data;
  new_node->next= (*head_ref);
  (*head_ref)= new_node;
}
```

```
int main()
{
    struct Node* head=NULL;
    Push (&head, 12);
    Push (&head,29);
    Push (&head, 11);
    Push (& head, 23);
     Push(&head, 8);
    Print node (head);
     return 0;
}
```

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

Ans The major difference blw 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 relies on references to the previous and next element.

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

```
#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 printlist(struct node *head)
{
    struct node *temp=head;
    while (temp!=NULL)
    {
        printf("%d", temp->data);
        temp= temp->next;
    }
    printf("\n");
}
```