① write a programme to insert and delete an element at the nth and kth position in a linked list where n and k are taken from user.

```
#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 < p-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, p;
head = NULL;
printf ("Enter the position for and inserting
  ...");
scanf ("%d", &n);
scanf ("%d", &x);
Insert
  insert (x, n);
  printf ("Enter the position to delete");
  scanf ("%d", &k);
  delete (k);
  printf ("");
  return 0;
```

(2) Construct a new linked list by merging alternative nodes and add them to a new list.

Example in linked list 1 we have {1,2,3} and in the second list {4,2,6} list 2 and in the new we should have {1,4,2,2,3,6}.

```
#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
                    (size of (struct node))
new -> data = data ;
new -> next = * head;
* head = new ;
}
```

```
struct node * merge (struct node *a, struct node
                     *b)
{
    struct node false;
    struct node *tail = false;
    false.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 false.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 (&a, key[i]);
for (int i = n-2; i>=0; i=i-2)
    push (&b, key[i]);
struct node *head = merge (a, b);
printf list (head);
}
```

(3) Find all the elements in the array whose sum is equal to s (where s is given by the user)

```
#include <stdio.h>
void find (int arr[], int n, int s){
    int sum = 0;
    int l=0, h=0;
    for (l=0; l<n; l++){
        while (sum<s, l<h<n)
            sum += arr[h];
            h++;
        if (sum == s)
        {
            printf ("found");
            return;}
        sum -= arr[l];
    }
}
int main (void){
    int arr[] = {2,6,0,9,7,3}
    int s = 15;
    int n = size of (arr) / size of (arr[0]);
    find (arr, n, s);
    return 0;
}
```

④ write a program to print the elements in a queue:

(i) in reverse order (ii) in alternate order

```
#include <stdio.h>
#include <stdlib.h>
struct node {
        int data
        struct node *next;
};
void printf rev (struct node *head){
    if (head == NULL)
        return;
        printf rev (head -> next);
        printf ("%d" head -> data);
        printf (
void push (struct node * head rev, clar new
{
    struct node _new = (struct node*) malloc
                    (sizeof (struct node));
    node _new -> data = new;
    node _new -> next = (head * _ref);
    (* head _ref) = node _new;
}
    int main ( )
        struct node * head = NULL;
```

```
    push (&head, 4);
    push (&head, 3);
    push (&head, 2);
    printf new (head); pr
    printf alternate (head);
    return 0;
}
```

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

~~The~~

Sol The major difference between array and linked list regards to their structure.

Arrays are index based data structure where each element associated with an reference to the previous and next element

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

~~# include <st~~

Ans

```
#include <stdio.h>
# include <stdio.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;
    }
    printf ("\n");
}
void merge (struct node *p, struct node *q)
void merge (struct node *p, struct node **q)
{
    struct node * p_curr = p, *q_curr = *q;
    struct node *p_next, *q_next;
    {
        p_next = p_curr => next;
        q_next = q_curr => next;
        q_next => next = p_next;
```

```
p->curr = p->next = q->curr;

p->curr = p->next;
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");
printlist(p);
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");
printf(p);
printf("modified second linked list:\n");
printlist(q);
getchar();
return 0;
```