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 the user.

```c
#include <stdio.h>
#include <stdlib.h>
void ans (node *, int, int)
int size = 0;
struct node {
int data;
struct node * next;
}
node * get node (int data)
{
    node * newnode = (struct node *) malloc (newnode);
    newnode → data = data;
    newnode → next = null;
    streturn newnode;
}
void ins (node * current, int pos, int data)
{
  if (pos<1 || pos > size+1)
  printf ("Invalid");
  else
  {
    while (pos--)
    {
      if (pos==0)
      {
        node * temp = get node (data);
        temp → next = * current;
        * current = temp;
      }
    else
    {
    current = &(* current) → next;
    }
    size++;
```

```c
    }
}

void printl( struct node * head)
{
    while ( head != null)
    {
        printl ("/d", head -> data);
        head = head -> next;
    }
    printl ("\n");
}
void del ( struct node * head uef, int pos) {
    if ( head_ uef == null)
    return;
        temp = head_ uef;
    if ( pos = 0)
    {
        * head- uef = temp -> next;
        free (temp);
        return;
        for (int i = 0; templ = NULL && 7 < pos -1; i++)
        {
            temp = temp -> next;
            free ( temp -> next);
            temp -> next = next;
        }
    }
    int main()
    {
        struct node * head = NULL;
        Push ( & head, 9);
        Push ( & head, 8);
        Push ( & head, 5);
        ins ( & head, 9, 4);
        del ( & head, 6);
        Print list (head);
        return (0);
    }
```

② construct a new linked list by merging alternate 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 we should have {1, 4, 2, 5, 3, 6, }.

```c
# include < stdio.h >
# include < stdlib.h >
struct node {
    int data;
    struct node * next;
}
void PrintList (struct node * head)
{
    struct node * Ptr = head;
    While (Ptr)
    {
        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 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;
        }
        else
        {
            tail → next = a;
```

```c
        tail = a;
        a = a->next;
          tail->next = b;
       }
    }
    return dummy.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-2)
       Push(&a, keys[i]);
   for (int i = n-2; i >= 0; i = i-2)
       Push(&b, key[i]);
   struct node *head = merge (a,b);
   Printlist (head);
}
```

② Find all the elements in the stack whose sum is equal to K
(where K is given by the user).

```c
# 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 && h < n)
    sum += arr[h];
    h++;
if (sum == s)
}
    printf ("found");
   return;
}
   sum -= arr[l];
}
}
int main (void)
{
```

```c
int arr[] = {2, 6, 10, 9, 9, 3}
int s = 15;
int n = size of (arr)/size of (arr[0]);
find (arr, n, s);
return 0;
}
```

(11) While a program to print the elements in a queue.
i) in reverse order        ii) in alternate order.

```c
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node *next;
}
void print rev(struct node * head)
{
    if (head == NULL)
        return;
    print rev(head → next);
    printf("%d", head → data);
void Push(struct node * head rev, char new)
{
    struct node * node_new = (struct node *) malloc(size of (struct node))
    node_new → data = new;
    node_new → next = (head *_ref);
    (*head-ref) = node_new;
}
int main()
    struct node *head = NULL;
    Push(&head, 2);
    Push(&head, 4);
    Push(&head, 6);
    Print new(head); Print alternate(head);
    return 0;
}
```

```cpp
void print alternate (struct node * head)
{
  int count = 0;
  While (head! = NULL)
  {
    if (count %2 == 0)
        count << head → data << " ' ';
        Count ++;
        head = head → next;
  }
}
```

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

Answer:-

Key differences between Array and linked list

1) An array is a data structure that contains a collection of similar type data elements whereas the Linked list is Considered as non-primitive datastructure Contains a Collection of Unordered linked elements known as nodes.

2) In the {elements} array the elements belong to indexes, i.e., if you want to get into the fourth element you wohave to write the variable name with its index of location within the square bracket.

3) In a linked list through, you have to start from the head and walk your way through until you get to the fourth element.

4) Accessing an element in an array is fast while in linked list takes linear time, so it is quite a bit slower.

5) Operations like insertion and deletion in array Consume a lot of time. On the other hand the performance of these operations in linked list is fast.

6) In a array, memory is assigned during Compile time while in linked list it is allocated during execution of runtime.

```c
# include <stdio.h>
# include <stdlib.h>
int len (int a[])
{
   int i=0, a m=0;
   while (1)
   {
      if (a[i])
      {
         an++, i++;
      }
      else
      {
         break;
      }
   }
   return an;
}

Void changinglist (int a[a], int b[])
{
   for (int i= len(a)-1; i>=0; i--)
   {
      a[i+1] = a[i];
```

```c
a [o] = b[0];
Printf("\n the elements of first array: \n");
for (int i =0; i<len(a); i++)
{
    Printf("%d", a[i]);
}
for (int i=0; i<len(b); i++)
{
    b[i]=b[i+1];
}
Printf("\n the elements of second array: \n");
for (int i=0; i<len(b); i++)
{
    printf("%d", b[i]);
}
}
int main()
{
    int a [10]= {1,2,3}, b[10] = {4,5,6};
    changing list (a,b);
}
```