1. Write a program to insert and delete an element at The nth and kth pointer in a linked list where n and k are taken from the users

```
#include< stdio.h>
# include< stdlib.h >
struct Node {
int data;
struct Node *next;
};
struct Node * head;
void ins (int data, int n)
{
    node * temp = newnode();
    temp → data = data;
    temp → next = NULL;
    if (n==1)
    {
        temp → next = head;
        head = temp;
    }
    return;
}
void del( int k)
{
    struct Node * temp = head;
    if (k==1)
    {
        head = temp → next;
    free (temp);
     return;
    }
    Node *temp = head;
    ~~for~~ { ~~int~~
    int i;
    for (i=0 ; i<n-2; i++)
    {
         temp = temp -> next;
    }
     temp -> next = temp -> next;
     temp -> next = temp;
}
void  print ();
~~for~~ {
 int i;
 for( i=0; i< k-2; i++)
 {
      temp = temp -> next;
      free (temp);
  }
  int main()
  {
     int n,a,k;
     head = NULL;
     printf(" Enter the position for insertion");
    scanf("%d",&n);
    scanf("%d",&a);
    ins(a,n);
    printf(" Enter the position to delete");
    scanf("%d", &k);
    del(k) ;
    print(a);
    return ;
}
```

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 list 2 we have {4, 5, 6} in the new list we should have {1, 4, 2, 5, 3, 6}.

Ans:-

```
#include< stdio.h>
# include< stdlib.h>
struct node
{
    int data;
    Struct node *next;
};
void push( struct node ** headref, int newdata)
{
    struct node* newnode = (struct node *) malloc (sizeof(
                                        struct node));
    newdata → data = newdata;
    newdata → next = (* headref);
    (* headref) = newnode;
}
```

```
void print (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)
{
    struct node * Pcurr = P, * qcurr = *q;
    ~~struct node * pcurr =~~
    struct node * pnext, * qnext;
    while ( pcurr != NULL && qcurr != NULL)
    {
        pnext = pcurr -> next;
        qnext = qcurr -> next;
        qcurr -> next = pnext;
        pcurr -> next = qcurr;
        pcurr = pnext;
        qcurr = qnext;
    }
    *q = qcurr;
}
```

```
int main()
{
    struct node *p = NULL, *q = NULL;
    push(&p, 3);
    push(&p, 2);
    push(&p, 1);
    printf(" first linked list:\n");
    print(p);

    push(&q, 8);
    push(&q, 7);
    push(&q, 6);
    push(&q, 5);
    push(&q, 4);
    printf(" second linked list:\n");
    print(q);

    merge(p, &q);

    printf(" Modified first linked list:\n");
    print(p);
    printf(" Modified second linked list:\n");
    print(q);
    getchar();
    return 0;
}
```

Output:-

```
first linked list :
1 2 3

second linked list:
4 5 6 7 8

Modified first linked list :
1 4 2 5 3 6

Modified second linked list:
7 8
```

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

Ans:-

```
#include<stdio.h>
 int top = -1;
  int x;
 char stack[100];
 void push(int x);
 char pop();
 int main()
 {
     int i, n, a, t, k, f, sum=0, count = 1;
    printf(" Enter the number of elements");
    scanf("%d", &n);
    for(i=0; i<n; i++)
    {
        printf(" Enter element");
    scanf ("%d", &a);
    push(a);
 }
 printf(" Enter sum to be checked");
 scanf("%d", &k);
 for(i=0; i<n; i++)
 {   int j;
     t = pop()
     sum += t;
     count += 1;
     if( sum= =k){
     for (j=0;j< count ; j++)
     printf("%d", stack[j]);
     f =1 ;
     break;
     }
     push(t);
 }
 if (f!=1)
  printf ("The elements in The stack dont add up");
 }
 void push(int x)
  {
     if (top = = 99)
     {
        printf(" stack is full\n");
        return;
     }
```

```
        top+= ~~to~~1;
        Stack[top] = x;
}
char pop ()
{
        if (stack [top] == -1)
        {
                printf(" Stack is empty \n");
                returno;
        }
        x = stack[top];
        top = top -1;
        return x;
}
}
```

output:-

```
Enter number of elements 5
Enter element 1
Enter element 2
Enter element 3
Enter element 4
Enter element 5

Enter the sum to be checked 5
23 14
```

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

i Reverse order.

ii alternative order.

Ans:-

```
# 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
                            (sizeof (struct node));
    node_new → data = new;
    node_new → next = (*headref);
    (* headref) = node_new;
}
```

```
int main()
{
    struct node * head = NULL;
    push (& head, 4);
    push (& head, 3);
    push (& head, 2);
    printnew ( head) ; print alternative (head);
    returno;
}

void printalternative (struct node * head)
{
    int count = 0;
    while ( head! = NULL)
    {
        if ( count % 2 = =0)
        {
            count << head → data << " ";
            count ++;
            head = head → next;
        }
    }
```

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

Ans:-

The main differences between Array and linked list are.

① An array is a data structure that contains collections of similar type of data elements.

whereas the linked list is considered as non primitive data structure contains a collection of unordered linked elements known as nodes

(2) In the array the elements belong to indexes

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

(4). Accessing the elements in an array is very fast when compared to linked list.

(5) In array memory is assigned during compile time whereas in linked list it is allocated during execution time.

(ii)

```
# include < stdio.h>
# include< std lib.h>
int len (int a[ ])
{
 int i=0, an=0;
  while (1)
  {
     if (a[i])
     {
        an++, i++;
     }
```

```
        else
        {
                break;

        }
    }
    returnan;

}
void changelist ( int a[ ], int b[ ])

{   int i;
    for ( ~~int~~ i= len (a) - 1; i>=0; i--)

    {
        a[i+1] = a[i];

    }

    a[0] = b[0];

    printf("The elements of first array\n");

    for ( i =0; i< len (b); i++)

    {
          b[i] = b[i+1];

    }
    printf(" The elements of second array\n");

    for ( i =0; 1< len(b) ; i++ )

    {
          printf(" %d", b[i])

    }
}
```

```
int main()
{
    int a[10] = {1, 2, 3} , b[10] = {4, 5, 6};
    change list (a,b);
}
```

Output:-

```
The elements of first array     1,2,3

The elements of second array    4,5,6

4,1,2,3

 5,6
```