① Wap to insert and delete an element at the n^{th} and k^{th} position in a linked list where n and k is taken from user.

A)
```
#include<stdio.h>
#include<stdlib.h>
struct node
{
 int data;
 structNode *next;
};
 Struct Node *head;
 Void Insert(int data, int n){
 Node *temp = new node n;
 temp -> data = data;
 temp -> next = null;
 if (n == 1){
 temp -> next = head;
  head = temp;
  return;
 }
 void data - (int k){
 struct Node *temp = head;
 if (k = 1){
 head = temp -> next;
 free (temp);
 return
 }
 Node *temp = head;
 for (int i == 0, i < n-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, k;
head = null;
printf ("Enter the position for inserting");
Scanf ("%d" &n);
scanf ("%d", &x);
Insert (x, n);
printf ("Enter the position to delete");
Scanf ("%d", &k);
Delete (k);
print (x)
return;
}
```

(2) Construct a new linked list by merging alternative nodes and two lists for example in list1 we have {1, 2} { in list 2 we have {4,5, 6} ; in the new list we have {1, 4, 2, 5, 3, 6}

A)

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node *next;
}
void printlist(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 *) mallo
                (sizeof(Structnode));
new->data = data;
new -> next = *head;
 *head = new;
}
Struct node *merge(struct node *a, struct node *b)
{
strait node fake;
Structnode *fail = fake;
fake. next = Null;
whi le(1){
if (a == Null)
{
fail ->next = a;
break;
```

```
}
else if (b = Null)
{
fail -> next = a;
break;
}
else
{ tail -> next = a
  fail = a;
  a = a -> next;
  tail -> next - b;
  }
}
return fake.next;
}
void main()
{
  int key[] = {1, 2, 3, 4, 5, 6, 7}
  int n = sizeof(key)/sizeof 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)
  1    Push(&b ; key[i]);
  Struct node *head = merge(a,b);
  print list (head);
}
```

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

```
#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 =0;
 printf("Enter no of elements in stack");
 scanf ("%d", &n);
 for(i=0 ; i<n; i++){
  printf("Enter next element");
  scanf("%d", &a);
  push(a);
 }
 printf("Enter the sum to be checked");
 scanf(" %d ", &k);
 for(i=0;i<n;i++)
 {
  t = pop();
  sum+ = t;
  count+ = 1;
  if(sum == k) {
   for(int j=0; j<count; j++)
   printf("%d", stack[j]);
   f = 1;
   break;
  }
  push(t);
 }
 if (f != 1)
 Printf("The element is stack dont add up to sum");
}
 void push(int x)
{
 if(top == 99
 {
```

```
    printf("\nstack is Full!!!\n");
    return;
  }
  top = top + 1;
  stack[top] = x;
}
char pop()
{
  if (stack[top] == -1)
  {
    printf("\nstack is Empty!!!\n");
    return 0;
  }
  x = stack[top];
  top = top - 1;
  return x;
}
```

④ Wap to print the elements in a queue
i) Reverse order
ii) Alternate order

A]
```
#include <stdio.h>
#define SIZE 10
void insert (int);
void delete();
int queue[10], f=-1, r=-1;
void main(){
    int value, choice;
    while(1){
        printf("\n\n***MENU***\n");
        Printf("1. Insertion\n2. Deletion\n3. Print Reverse\n4. Print Alternate\n5. Exit");
        printf("Enter your choice :");
        Scanf("%d", choice);
        switch(choice){
case 1: printf("Enter the value to insert");
        scanf("%d", &value);
        insert(value);
        break;
case 2: delete();
        break;
```

```
case 3 : printf("Reversed queue is: ");
         for(int i=SIZE; i>=0; i--)
{
         if (queue[i] == 0)
         continue;
         printf("%d", queue[i]);
}
         break;
case 4 : printf("Alternate elements of queue")
         for(int i=0; i<size; i+=2)
{
         if(queue[i] == 0)
         continue;
         printf("%d", queue[i]);
}
         break;
case 5 : exit(0);
default: printf("\n Wrong selection!!!");
         }
}
void insert(int value){
         if ((f==0 && r==SIZE-1) || f==r+1)
           printf("\nQueue is Full");
         else{
           if(f==-1)
f=0;     r=(r+1)%SIZE;
         queue[r] = value;
         printf("\n Intersection successful!!!");
}}
```

```
void delete(){
    if (f==-1)
        Printf("\nQueue is empty!!!");
    else {
        printf("\n Deleted : %d ",queue[f]);
        f = (f+1)%SIZE;
        if (f==r)
    f=r= -1;
}}
```

5 (i) How array is different from Linked List

Ans The main difference b/w Array & Linked list regards to their structures.
Arrays are index based data structure where each element associated with an indesl. On the other hand, Linked List relies on references to the previous and next element.

(ii) Wap to add the first element of the List to a another list for example we heave {1,2,3} in List2 we have to get {4,1,2,3 as output for list1 and {5,6} for list2

Ans

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node* neset;
};
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)
{
    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");
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");
Print List(p);
Printf("Modified Second Linked list : \n");
Print List(q);
getchar();
return 0;
}
```