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 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 = neco 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(inti=0; i<n-2; i++){
  temp= temp->next;
  }
  temp->next = temp->next;
  temp->next = temp;
  }
   void print();
```

```
for (int i=0; i<(c-2); i++)
  temp = temp -> next;
  free(temp);
  }
int main(){
int n, x, k;
head = NULL;
printf("enter the position for and 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 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}

```
#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 fake;
  struct node * tail = fake;
    fake.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 fake.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);
 print list (head);
}
```

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

```
# include < stdio.h >
int top = -1;
int x;
char stack [100];
void push (int x);
char pop();
int main()
{
int i, n, a, b, k, f, sum=0, count=1;
print f ("enter the number of elements in the
                                    stack");

scan f ("%d", &n);
for ( i=0; i<n; i++){
print f ("enter next element");
 scan f ("%d", &a);
 push (a);
```

```
{
printf("Enter the sum to be checked");
scanf("%d",&k);
for(i=0; i<n; i++)
{
t=pop();
sum+=t;
coont+=1;
if(sum==k){
for(int j=0;j<coont;j++)
printf("%d",stack[j]);
f=1;
break;
}
push(t);
}
if(f!=1)
printf("The elements in the stack dont add up to
                the sum");
}
void push(int x)
{
if(top==99)
{
printf("\n stack is FULL!!!\n");
return;
}
top=top+1;
stack[top]=x;
}
char pop()
{
if(stack[top]==-1)
{
printf("\n stack is EMPTY!!!\n");
return 0;
```

```
}
x = stack[top];
top = top-1;
    return x;
}
```

4. Write a program to print the elements in a queue.
   i. in reverse order
   ii. in alternate order

i.

```
#include <stdio.h>
#include "stack.h"
#include "QQ.h"
int main()
{
  int n, arr[20], i, j=0;
    struct stack s;
    init stack (&s);
    printf("enter no");
    scanf("%d", &n);
   for( i=0; i<n;i++)
   {
     printf("enter values : ");
     scanf("%d", &arr[i]);
   }
   for( i=0; i<n; i++)
   {
     insert (arr[i]);
   }
     while( j!=n)
     {
       push(&s, del());
       j++;
```

```
    }
    print ("Reverse is");
    while (stop != -1)
    {
     printf("%d", pop(&s));
    }
    printf("\n");
  return 0;
}
```

ii.

```
#include <stdio.h>
#include <stdlib.h>
 struct node {
   int data;
   struct Node * next;
  }
   void print nodes ( struct node * head)
   {
    int count = 0;
    while (head != NULL) {
        if (coount % 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 many array is different from the linked list
ii. Write a program to add the first element of one list to 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.

i) The Major difference between 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 reference to the previous and next elements.

ii)

```
#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 (sizeof (struct node));
 new_node -> data = new_data;
 new_node -> next = (& head_ref);
 (* head_ref) = new_node;
}
void print list(struct node * head)
{
 struct node * temp = head;
 while (temp! = NULL)
 {
 printf("%d", temp -> data);
 temp = temp -> next;
 }
 printf("\n");
}
```