1. Write a program to insert and delete an element at the n^{th} and K^{th} position in a linked list where n and K are taken from user.

Sol:

```
#include <stdio.h>
#include <stdlib.h>
Struct node {
int data;
Struct node * neset;
};
Struct node * head;
Void Insert (int data, int n) {
Node * temp = new node();
temp-> data = data;
temp -> neset = Null;
if (n==1) {
temp -> neset = head;
head = temp;
return;
}
Void delete (int R);
Struct node * temp = head;
if (R==1) {
head = temp -> neet;
free (temp);
retvan;
}
Node * temp = head;
```

Teacher's Signature :

Date

Expt. No. Page No.

```
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<R-2; i++)
  temp = temp -> next;
  free (temp);
}
int main {
int n, x, R;
head = Null;
Print F ("enter the position to insert:");
scanf ("%d", &n);
Scanf ("%d", & x);
Insert (x, n);
Print F (" Enter the position to delete:");
scanf ("%d", & R);
Delete (R);
Print (x,0);
return;
}
```

Date

Expt. No. _____ Page No. _____

2 Construct a new linked list by merging alternate nodes of two lists.

Sol:

```
#include <stdio.h>
#include <stdlib.h>
Struct Node
{
  int data;
  Struct Node *next;
}
Void printlist (Struct Node* head)
{
 Struct * Node* ptr = head;
 while (Ptr)
 {
   print f ("%d -> ", Ptr -> data);
   ptr = ptr -> next;
 }
 print f ("Null\n");
}
Void push (Struct Node ** head, int data)
{
 Struct Node * new Node = (Struct Node*) malloc (size of (Struct node));
new Node -> data = data;
 new Node -> next = * head;
 * head = new Node;
}
```

Teacher's Signature : _____

```
Struct Node * shuffle merge (Struct Node *a, struct Node * b)
{
  Struct Node dummy;
  Struct Node * tail = & dummy;
  dummy . neset = Null;
  while (1)
  {
     if (a == Null)
     {
        tail -> neset = b;
        break;
     }
     else if (b == Null)
     {
        tail -> neset = a;
        break;
     }
     else
     {
       tail -> neset = a;
       tail = a;
       a = a -> neset;

      tail -> neset = b;
      tail = b;
      b = b -> neset;
     }
```

```
    return dummy.next;
}
int main (void)
{
  int Keys [] = {1, 2, 3, 4, 5, 6, 7};
  int n = size of (Keys) / sizeof (Keys [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 , Keys [i]);

  printf (" First list : ");
  printlist (a);

  Printf (" second list : ");
  Print list (b);

  Struct Node *head = ShuffleMerge (a,b);
  PrintF ("After merge : ") ;
  printlist (head);

    return 0;
}
```

Date

Expt. No. Page No.

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

Sol:-

```
#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 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 = t;
 count t = 1;
if (sum == K) {
for (int j=0; j<count; j++)
```

Teacher's Signature :

```
printf("%d", stack[i]);
f=1;
break;
}
Push(t);
}
if(f!=1)
printf("Elements in stack don't add up to 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];
```

Date

Expt. No. Page No.

```
    top = top-1;
    return x;
}
```

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

Sol:

```
#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\n 2. Deletion\n 3. Print Reverse\n
              4. Print Alternate\n 5. Exit");
    Printf ("\n Enter your choice:");
    scanf ("%d", & choice);
    switch (choice) {
          case 1:
          Printf ("Enter the value to be insert:");
          Scanf ("%d", & value);
          insert (value);
           break;
         case 2:
           delete ();
           break;
         case 3:
```

Teacher's Signature :

```
        Printf ("The 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 are:");
      for (int i=0; i<size; i+=2)
{
    if (queue[i] ==0)
    continue;
    Printf ("%d", queue[i]);
}
    break;
    case 5:
      exit (0);
    default:
      printf ("In Wrong selection!! Try again!!!");
    }
}}
Void insert (int value){
  if ((f==0 && r == size-1) || f==r+1)
    printf ("\nQueue is Full!! Insertion not possible!!!");
  else {
    if (f==-1)
```

Teacher's Signature :

```
f = 0;
    r = (r+1) % size;
    queue[r] = value;
    printf("\n Insertion success!!!");
}}
void delete() {
  if (f == -1)
    printf("\n Queue is empty!!! Deletion is not possible!!!");
  else {
    printf("\n Deleted : %d", queue[f]);
    f = (f+1) % size;
    if (f == r)
f = r = -1;
}}
```

5. i) How array is different from the linked list
ii) write a program to add the first element of one list to another list.

Sol:

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. whereas linked list relies on references where each node consists of the data and the references to the previous and next element.

ii)
```
#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;
}
```

Teacher's Signature : .....

Date

Expt. No. Page No.

```
    printf ("Null\n");
}
void push (struct Node **head, int data)
{
 struct Node* newNode = (struct Node*) malloc (sizeof (struct node));
 newNode -> data = data;
 newNode -> next = *head;
  *head = newNode;
}
void moveNode (struct Node** destRef, struct node** sourceRef)
{
 if (*sourceRef == Null)
   return;
 struct Node* newNode = *sourceRef;
 *sourceRef = (*sourceRef) -> next;
 newNode -> next = *destRef;
 *destRef = newNode;
}
int main (void)
{
  int keys [] = {1, 2, 3};
  int n = sizeof (keys)/sizeof (keys[0]);
  struct Node *a = Null;
  for (int i = n-1; i >=0; i--)
    push (&a, keys[i]);

 struct Node * b = Null;
 for (int i = 0; i<n; i++)
```

Teacher's Signature :

Date

Expt. No. Page No.

```
    push(&b, 2*keys[i]);

    moveNode(&a, &b);

    printf("First list: ");
    printList(a);

    Printf("second list: ");
    printList(b);

    return 0;
}
```

Teacher's Signature :