ASSIGNMENT-4

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 user.

A)
```
#include<stdio.h>
 #include<stdlib.h>
  struct Node {
   int data;
   struct Node* next;
   };
   struct Node* head;
   void Insert (int data, int n){
       Node* temp1= new Node ();
       temp1->data = data;
       temp1->next = NULL;
       if(n==1){
            temp1->next=head;
            head = temp1;
            return;
       }
       Node* temp2 = head;
       for(int i=0; i<n-2; i++){
               temp2 = temp2->next;
        }
        temp1->next= temp2->next;
        temp2->next = temp1;

    }
    void Print ();
    void Delete (int K){
```

```
struct Node* temp1= head;
if(k==1){
   head= temp1->next;
   free(temp1);
   return;

 }
 int i;
 for (i=0; i<k-2; i++)
  temp1=temp1->next;
  struct Node* temp2 = temp1->next;
   temp1->next = temp2->next;
   free(temp2);

}
int main(){
     head = NULL;
     Insert (2,1);
     Insert (3,2);
     Insert (4,1);
      Insert (5,2);
      Print ();
       int K;
        printf("Enter the position \n");
        scanf("%d", &K);
        Delete (K);
         Print ();
     }
```

2) #1 Construct a new linked list by merging alternate nodes of two lists.

```
#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* newNode = (struct Node*)malloc (sizeof( struct Node));
   newNode->data= data;
   newNode-> next =*head;
   *head = newNode;
    }
   struct Node* shuffle 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;
tail=a ;
a=a->next;
tail->next=b;
tail=b;
b=b->next;
}
}
return dummy.next;
}
int main(void)
{
int Keys[]= {1,2,3,4,5,6};
int n= sizeof(Keys)/sizeof(Keys[0]);
struct Node* a= NULL,*b= NULL;
```

```
for (int i=n-2 ; i>=0; i=i-1)
push(&a, Keys[i]);
for(int i=n-1; i>=3; i=i-1)
push(&b, Keys [i]);
printf("First List");
print List (a);
printf ("Second List");
print{List (b),
struct Node* head= ShuffleMerge(a,b);
printf(" AfterMerge"),
print List(head);
return 0;
}
```

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,a,t,k,f, sum=0, count=-1;
printf ("Enter the number of elements in the stack");
scanf ("%d" ,&n);
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 elements in the 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];
   top=top-1;
       return x;
  }
```

4) Write a Program to print the elements in a queue.
1) in reverse Order.
2) in Alternate Order.

```
#include<stdio.h>
#define SIZE 10
void insert (int);
void delete ();
int queue[10], fd=-1, rt=-1;
void main(){
    int value, choice;
    while(1){
        printf("\n\n*** MENU*** \n");
        print f (" 1. Insertion \n 2. Deletion \n 3. print Reverse \n 4. Print Altornate
                \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: 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 Order (elements) of the 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 ("\n wrong Selection!! Try again");
             }

        }
        }
    void insert (int value){
        if ((d==0) & t == SIZE-1) || d= t+1)
              printf ("\n Queue is full");
           else {
             if (d==-1)
                  d = 0;
                  t = (t+1) % SIZE ;
                  queue [t] = Value ;
                  printf ("\n Insertion Success");

             }
             }
      void delete(){
          if (d==-1)
              printf ("\n Queue is Empty ");
           else {
             printf (" \n Deleted : %d", queue [d]);
             d = (d+1) % SIZE;
             if (d==t)
                  d=t=-1
             }
             }
```

Scanned with CamScanner

5)
1) How array is different from linked list.

A) The Major difference is variance between structure in both of them.

Arrays are index based data structure where each element associated with an index.

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

Array is a set of similar data Objects stored in a sequential memory.

Linked list is a data structure which contains a sequence of each element is linked to it's next element.

2) Write a program to add the first element of one list to another list

```
#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 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,3);
    push (&p,2);
    push (&p,1);
    printf(" First Linked List :\n");
    printList(p);
    push(&q,6);
    push(&q,5);
    push(&q,4);
    printf("Second Linked List \n");
    printList(q);
    merge(p,&q);
    printf(" Modified First Linked List :\n");
    printList(p);
    printf(" Modified Second Linked List :\n");
    printList(q);
    getchar();
    return 0;
}
```