1. ASSIGNMENT

K.GEETIKA
AP19110010123
CSE-G

Write a program to insert and delete an element at the nth and kth position in a linked list where n and k ps taken from user.

Program:-

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
int data;
struct node* next;
};
struct node *head;
void Insert (int data, int n) {
Node *temp = new node;
temp->data = data;
temp->next = null;
if (n==1) {
temp -> next = head;
head = temp;
return;
}
void Delete - (Pnt k) {
struct Node * temp = head;
if (k == i) {
head = temp -> next;
free (temp);
return;
}
node * temp = head;
for (int i=0; i<n-2, it+) {
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 and inserting :");
 Sconf ("%d" &n);
 Scanf ("%d", &x);
 insert (x, n);
 Print ("enter the position to delete)
 Scan f ("%d", &k);
 Delete (k);
 Print (x);
 return;
 }
```

2. construct a new linked list by merging alternative nodes and two lists for example in list 1 we have {1,2} and list 2 {4,2,6} and in the new we should have {1,4,2,5,3,6}

Program:-

```
#include <stdio.h>
#include <stdlib.h>
Structnode{
 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) malla
        (size of (structnode));
  new →data = data;
  new → next = *head;
 *head=new;
 }
 structnode * 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;
  }
   trail-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-i, i>0; i=i-a)
  Push(&a, key[i]);
for(Pnt i=n-2; i>=0; i=i-2)
  Push(&b; key[j]);
struct node* head =merge(a,b);
PrintList(head);
}
```

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

Program:

```
#include<stdio.h>
Pnt top=-1;
Pnt x;
char stack[100];
Void Push(int x);
char pop();
Pntmain()
{
Pnt i,n,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++){
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++)
Print("%d", stack[j]);
f=1;
break;
}
Push(t);
}
if(f!=1)
Printf("The elements in the stack don't 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 ("\nstack 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.

Program:-

```
#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){
  Print("\n\n*** MENU***\n");
  Print("1. Insertion\n2. Deletion\n3. Print Reverse\n4.
                                    Print Alternate\n5. EXIT");
  Print("\n Enter your choice:");
  scanf("%d", &choice);
  switch(choice){
```

```
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++)
Print("%d",stack[j]);
f=1;
break;
}
Push(t);
}
if(f!=1)
Printf("The elements in the stack don't add up to the sum");
}
Void push(int x)
{
if(top==99)
{
Printf("\n stack is Full !!!.\n");
return;
}
```

```
case1: Printf ("enter the value to be insert :");
       scanf ("%d", &value);
       insert (value);
       break;
       case 2 : delete();
       break;
       case3 :
            Print (" The reversed queue is;");
            for (int i=SIZE ; i>=0; i--)
{
            if (queue[i] ==0)
            continue;
            Printf ("%d", queue[i]);
}
        break;
       case4 :
            Print f("Alternate elements of the queue are:");
            for (int i=0; i<SIZE ; i+=2)
{
        if (queue [i]==0)
        continue;
        Print ("%d", queue[i]);
}
      break;

case5 : exit(0);
default : Printf ("\n wrong selection!!! Try again!!!");
    }
  }}
void insert (int value){
 if (f==0 && r=SIZE -1) || f==r+1)
```

```
Print ("\n Queue is Full!!! Insertion is not possible!!!");
else {
  if (f == -1)
f = 0;
  r = (r+1) % SIZE;
  queue[r] = value;
  Printf("\n Insertion Success!!!");
}}
void delete(){
 if (f == -1)
  Print ("\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.

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 references to the previous and next element.

(ii) WAP 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.

Program :-

```
#include <stdio.h>
#include <stdlib.h>
Data structure to store a linked list node
struct node
{
    int data;
    struct node * next;
}
void PrintList (struct node * head)
{
    struct node * ptr = head;
    while(ptr)
    {
        Print("%d -> ", ptr -> data);
        ptr = ptr -> next;
    }
    Printf("NULL\n");
}
void push(struct node ** head, int data)
{
    struct node * newnode = (struct node*) malloc(sizeof
    newnode -> data = data;                    (struct node));
    newnode -> next = * head;
    * head = newNode;
}
Void moveNode(struct node ** destRef, struct node **
                                        SourceRef)
{
    if(*SourreRef == NULL)
      return;
    struct node * newnode = * SourceRef;
    * SourceRef = (*SourreRef) -> next;
```