Assignment-4

Nandini.ch
AP19110010333
CSE-F

1. Write a program to insert and delete a 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 = newnode();
temp→ data = data;
temp → next = Null;
 if (n==1) {
  temp → next = head;
  head = temp;
  return;
 }
 void delete_int(){
  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",&x);
 scanf ("%d", &x);
 insert(x,n);
 printf ("Enter the to delete");
 scanf("%d",& k);
 Delete(k);
 printf(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 {1,2,3,4,5,6}

```
#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:
}
```

Scanned with CamScanner

```
tail→next=a;
 break;
 tail=a;
 a=a→next;
 tail→ next=b;
 }
}
 return fakenext;
}
 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-a)
     push(& a, keys[i]);
  for(int i=n-2; i>=0; i=i-2)
     push(& b; key [j]);
  struct node * head= merge(a, b);
  printf (head);
 }
```

3. Find all 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 a);
char pop()
int main()
```

Scanned with CamScanner

```
{
 int i, n, a, k, t, f, sum = 0, count = 1;
 printf("Enter the no. of element 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 += f;
  if (sum == k) {
 for (int j = 0; j < count; j++)
  printf("%d", stack[j]);
   break;
 }
 push(t);
 {
 if(f != 1)
   printf("The elements in stack don't add upto 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]=x;
}
 char pop ()
{ if (stack [top]==-1)
{
 printf("\stack empty");
 return 0;
}
 x = stack [top];
 top = top -1
}
```

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

```
#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 ** Menue**\n");
     printf("1. Insertion \n 2. Deletion \n 3. Reverse\n
                                              4. Alternate);
     printf("\n Enter your choice");
    Scanf (" %d", &choice);
     Switch(choice) {
Case(i) :- printf(" Enter the value to be inserted : ");
           Scanf (" %d", &value);
           insert (value);
           break;
case (ii) :- delete ()
             breaks;
case (iii) :- printf("The reversed queue is" )
              for (int i= SIZE; i >=0; i--)
 {
  if (queue[i] = =0)
  continue;
  printf("%d", queue[i]);
 }
  break;
case (iv) : printf("Alternate elements of queue)
            for (int i=0; i < SIZE ; i+ =2)
{
 if (queue [i] = = 0)
  continue;
 printf ("%d", queue [i]);
}
 break;
case(r):- exit (0);
default : printf (" Wrong selection")
 }
 }
}
void insert (int value)'
{
 if (if==0 && r= size -1) nf==r+1)
   printf("\n Queue is full")
   else{
    if (f==-1)
    f =0;
      r=(r+1) % size;
      queue[r] = value;
      printf("\n insertion success. ");

 }
}
void detete();{
 if (f== -1)
  printf("\n Queue is empty");

 else{
   printf ("\n deleted : %d", queue[f]);
   f =(f+1) % SIZE;
   if (F==r)
  F = r=-1;
 }
}
```

Scanned with CamScanner

5, i) How array is different from the linked list

The major diff b/w array & linked list regards to their structure. Arrays are based data structure where the element associated with index.
On other hand, linked list relies on reference to the previous & next element.

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
                    (size of (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");
}
```