ASSIGNMENT-4

NAME:- K.ANUGNA SAI

ROLL NO:- AP19110010414

CSE-F

1) Write a Program to insert and delete an Element at the nth and kth position in a linked list where n and kis taken from 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 = new 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;
             return;
       }
          Node* temp=head;
       for (int i=0; i<n-2; i++) {
              temp = temp->next;
       }
        temp->next = temp->next;
         temp->next = temp;
```

#1

```
void print();
for (int i=0; i<K-2; i++) {
      temp =Temp -> next;
   }
 int main() {
      int n,x,K;
      head =null;
      printf("Enter the position for inserting:");
      scanf("%d", &n);
      Scanf("%d", &x);
      Insert (x,n);
      printf("Enter the position to delete");
      Scanf("%d", &k);
      Delete (K);
       Printf(x);
       return 0;
   }
```

(8) Construct a new linked list by merging alternative nodes of 2 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 we have {1,4,2,5,3,6}?

Sol:

```
#include<stdio.h>
#include<stdlib.h>
#include<assert.h>
 Struct node
 {
   int data;
   Struct node *next;
 };
void move node (struct node **x, Struct node **y);
 Struct node* Sorted merge (Struct node *a, struct node *b),
{
  Struct node dummy;
  Struct node *tail=&dummy;
  dummy. next= Null;
  while(1)
  {
  if(a==Null)
  {
   *y =newnode->next;
   newnode->next=*x;
    *x=newnode;
  }
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 *node)
{
 while (node != Null)
 {
   printf("%d", node->data);
   node = node->next;
  }
}
tail->next = b;
 break;
}
else if (b == Null)
{
tail ->next = a;
 break;
}
if (a->data <= b->data)
{
 move node (&(tail)->next), &a);
}
 else
 {  move node (&tail)->next, &b);
 }
 tail = tail->next;
}
return (dummy.next);
}
void move node (struct node** x, struct node**y);
```

```
{
    struct node *newnode = *y,
    assert (new node != Null);
int main ()
{
  struct node* res = null;
  struct node *a = Null;
  struct node *b = Null;
   Push(&a, 1);
  Push (&b, 2);
  push (&a,3);
  push (&b,4);
  push (&b, 5);
  push(&b, 6);
  res = sorted merge (a,b);
  printf ("merge linked list is :\n");
  print list(res);
  return 0;
}
```

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 s1[10], top1= -1, s2[10] , top2= -1;
 int s1empty()
{ if(top1==-1)
        return 1;
else
    return 0;
}
int s1 top()
{
return s1[top 1];
}
int s1pop()
{ top 1--;
}
int s1 push(int n)
{ s1[++top1]=n;
}
int s2 empty()
{
 if (top2==-1)
     return 1;
else
    return 0;
```

```
}
int s2 top()
{
 return s2[top2];
}
 int s2 pop()
{
 top2--;
}
int s2 push(int n)
{
 s2[++top2]=n;
}
  int sum(int k)
{
  int n;
  while (s1 empty() != 1)
  {
    n = s1top();
    s1 pop();
   while (s1 empty ()!=1)
   {
     if (n+s1top()=k)
    {
     printf("%d, %d\n", n, s1top());
    }
    s2 push(s1top());
    s1 pop();
   }
   while (s2 empty()!=1)
  {
    s1 push (s2top());
    s2 pop();
```

```
            }
        }
    }
int main ( )
{
  int n, i, K;
  printf ("Enter The no.of Elements of stack :\n");
  scanf ("%d", &n);
  for (i=0; i<n; i++)
  {
    Scanf ("%d", &e);
    S1push(e);
  }
  printf ("Enter the value of Constant Sum:\n");
  Scanf ("%d", &k);
  printf( "The combinations whose sum is equal to k is:\n");
  Sum (K);
}
```

(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], k=-1, j=-1;
void main()
{
    int value, choice;
    while(1){
        printf("\n\n **MENU** \n");
        printf("1.Insertion \n 2.Deletion \n 3. Reverse order \n4.alternate
                \n 5.Exit);
        printf ("Enter your choice");
        Scanf("%d", &choice);
        Switch(choice) {
      Case1:- printf ("Enter the value to be insert:");
            Scanf( "%d", &value);
            insert(value);
            break;
     Case2:- 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 Elements to 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);
   void insert(int value){
         if (f==0&&r==size-1)|| f==r+1)
          printf ("\nqueue is full);
       else {
            if (f==-1)
       ~~f=0;~~ 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");
```

A+:

```
else{
  printf ("\n deleted : %d", quue [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 of 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 list1 and {5,6} for list2?

Sol: (i) The major difference b/w Array and linked lists regards to their structure, Arrays are index based ~~also~~ data structure where each element associated with an index on the other hand, linked list ~~either~~ relies on reference to the previous and 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");
}
```