DSA Assignment - 4

B.Sridhar
AP19110010447
CSE-F.

1) Write a program to insert and delete an element at nth and kth position in a linked list where n, K is taken from user.

```
#include <stdio.h>
#include <stdlib.h>
  Struct Node {
    int statistics {
    Struct Node * next;
  };
  Struct Node * head;
  Void Insert (int statistics , int a) {
    Node * temp = new node ();
    temp -> data = data;
    temp -> next = Null;
     if(a==1){
     temp -> next = head;
       head = temp;
       return;
   }
   Void Delete (int b) {
     Struct Node * temp = head;
     if(b== 1) {
     head = temp -> next;
     free (temp);
       return;
   }
     Node * temp = head;
for(int i=0; i<a-2; i++){
  temp= temp→next;
}
  temp→next =temp→next;
  temp→next =temp;
}
  Void Print ();
  for(i=0; i<b-2; i++){
    temp =temp→next;
      free(temp);
  }
  int main(){
  int a, b, x;
   head = Null;
   Printf(" Enter the Position for inserting: ");
   Scanf("%d", &a);
   Scanf("%. d", & x);
   Insert ( x, a);
    Printf(" Enter the Position fortodelete: ");
    Scanf("%d", b);
   Delete ( b);
   Print(x);
    return0;
   }
```

2) construct a new linked list by merging the alternate nodes of 2 lists for ex: in list 1. {1,2,3}, list 2. {4,5,6}.
New list may be {1,4,2,5,3,6}.

```
#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 * tail = wrong;
       wrong. next = Null;
     while (1) {
     if (a == Null) {
        tail → next = b;
          break;
     }
else if (b= Null) {
     tail→next=a;
     break;
  }
   tail→next=a;
       break;
  }
    else {
        tail→ next =a;
         tail = a;
         a=a→next;
          tail→ next-b;
     }
   }
    return wrong next;
   }

  Void main() {
   int Keys [] = {1,2,3,4, 5,6,7};
    int x = Sizeof (Keys) / size of key[0];
    Struct node *a= Null;  *b = Null;
    for(i= n-1; i>0; i=i-a) {
          Push (&a, keys(i]);
    for( int i=n-2 ; i>=0; i=i-2){
          Push (&b, keys[i]);
    Struct Node *head= Merge (a,b);
          Print list (head);
  }
```

3. Find all the program of elements in the Stack whose Sum is Equal to k.

```
# include <stdio.h>
 int top=-1;
  int x;
 char stack [100];
 void push (int a);
  char pop ();
 int main() {

  int  i, a, ~~x~~, t, b, f, sum=0, count=1;
  Printf(" Enter the no.of elements in the stack:");
  Scanf("%d", & x);
  for(i=0; i<x; i++){
      Printf(" Enter the elements to push in stack!);
      Scanf(" %d", &a);
       Push (a);
  }
 Printf(" Enter the Sum to be in the stack:);
  Scanf("%d", &b);
 for (i=0; i<x; i++){
     t= Pop ();
      Sum+ =t;
     count +=1;
     if (Sum==b) {
     for( j=0; j< count; j++){
         Printf("%d", stack (i));
          f=1;
          break;
     }
Push(t);
}
if(f!=1){
  Printf(" The elements inthe stack won't addup sum");
  }
  void Push (int a){
     if(top==99){
       Printf(" Stack is full" \n");
       return;
     }
     top=top+1;
     Stack[top]=a;
  }
   char Pop ()
  {
     if(stack(top])=[illegible]a;
  }
     char Pop ()
     if(stack(top])==-1
       Printf(" stack is Empty\n");
       return0;
     }
     x= stack [top];
         top= top-1;
     }
```

4)

i) reverse order:

```
struct queue info {
    int *array;
    int front;
     int rear;
     int space;
};
def struct queue info *queue;
  queue ( create queue (int max))
 {
     queue q;
    q = malloc( sizeof (struct queue info));
    if (q = Null) {
     Printf(" Error");
    q→ array = malloc (sizeof (int)*max);
    if (q→ array ==Null)
      Printf(" Error");
     ~~q→ array = malloc~~
     q→ space =max -1;
      q→ front =-1;
       q→ rear = -1;
       return q;
   }
   int is_full q(queue a) {
      return (q→ rear = q→ space);
   }
```

```
int is_empty(queue q){
    return (q→front = =-1);
  }
void enqueue (queue q, int x) {
  printf("underflow");
    return;
  }
 for (i=q→front; i<=rear; i++) {
 Printf("%d\t", q→array [i]);
 }
  intmain() {
  int max,x,i, choice,s,space, a,b,n=0;
   queue q;
   Printf("Enter the maxm elements");
    Scanf("%d", &max);
     q = createqueue(max);
     while(1) {
       Printf("\n Menu: 1. Insert 2. Display reversed in order
                    3. Exist");
      Printf(" Enter the choice");
       Scanf("%d", &choice);
       Switch (choice) {
         case 1:
       Printf("Enter the elements");
       Scanf("%d", &x);
       Enqueue (q, x);
          n++;
         break;
case 2:
Printf(" contents in the queue");
  display (q);
  for(i=0; i< Space ; i++){
     a= front and delete (q), s;
      Push(a, s);
   }
   q→front=-1;
   q→rear=-1;
   for (i=0; i< Space; i++){
     b= top and pop( s);
       Enqueue (q, b);
    }
   Printf("Reversed contentens are");
    display (q);
   case 3:
     Exist (0);
  }
 }
}
```

ii) alternate order?

```
#include < stdio.h>
#include <stdlib.h>
 struct node {
   int data;
   struct node* next;
 };
 void push (struct node* head ref, charnew) {
  struct node* node-new = (struct node*) malloc( sizeof(
                                               struct node));
   node.new → data =new;
    node_new →next =(*head_ref);
    (*head-ref)= node-new;
 }
  intmain() {
   struct node *head = NULL;
         push (&head,9);
         push(&head,19);
         push (&head, 29);
         push( &head, 39);
         push(& head, 0);
         printfaltnode(head) ;
         return0;
  }

  void print alternate (struct node *head) {
          int  count=0;
        while(head!= NULL) {
```

Scanned with CamScanner

```
if( count % 2==0){
    count <<head->data<<"  ";
     count ++;
    head=head->next;
  }
 }
};
```

5) i) Array consists of only single similar type of data, but the linked list consist of non-primitive data structure.

ii) Arrays belongs on the basis of indexes, and whereby accessing the element must be in the format of declaring the location, but in the linked list, we can by the head and node for the structure.

iii) The accessing element in array is faster whereas in the linked list, it takes a bit slower.

iv) Arrays are in the fixed size, but in linked lists, are dynamic and flexible, and increase its size.

v) The operations and like insertion, deletion in arrays consume a lot of time. But in linked lists, the Performance of these like operations are fast enough.

5(ii)

```
#include <stdio.h>
#include <stdlib.h>
 Structnode {
    int info;
    Struct node * next;
 };
 Void Push( Struct Node ** headref, int new_info) {
   Struct node* new-node = (Struct node*) malloc (sizeof
                                                  Structnode)
    new-node → data = new_info;
    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");
   }
```