Assignment-4

Ch. charankumar
AP19110010335, CSE-

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<malloc.h>
#include<stdlib.h>

struct node{
  int a;
  struct node*n
};
void insert();
void display();
void delete();
int Count();

type def structhode A_NODE;

A_NODE *head_node, *first_hode,
*temp_node =0, *prev_node, n_node;
int a;
```

Date :

Name of the Experiment :

Page No.:

```
int main(){
int selection=0;
Printf("only singly linked list");
while(selection<10)
{
 printf("selection");
 printf("1.Insert\n,2.Delete\n 3.Display\n,4.Count\n");
 Print f("Exit");
 printf("Enter any selection");
 Scanf("%d", &selection);
Switch(selection){
Case1:
insert();
 break;
Case2:
 deleteC);
 break;
 case3;
 displayc)
  break;
 Case4:
  Count();
  break;
  default:
  break;
}
```

```
Void insert ()
{
 Printf ("Insert linked list");
 Scanf ("%d", &A);
temp_node = (A_NODE*) malloc (sizeof (A_NODE));
 temp_node -> ~~val~~ a = A;
 if (first_node == 0)
 {
    First_node = temp_node;
}
else
   {
   head_node -> n = temp_node;
   }
   temp_node -> n = 0;
   head_node = temp_node;
    fflush(stdin);
   }
   void delete() {
   int Count a, position, c = 0;
    Count a = Count ();
```

```
temp_node = first_node;
Printf("Display linked list");
Printf("Enter deleted element position");
Scanf("%d", &position)
if(position>0 && position<=(counta)
{
if(position==1)
{
temp_node = temp_node->n;
first_node = temp_node;
Printf("Deleted");
}
else
 {
while(temp_node!=0)
{
if(c==(position-1))
{
prev_node->n = temp_node->n;
if(c==(counta-1))
{
head_node = prev_node;
}
Printf("Delete");
break;
```

```
        c++;
        Prev_node = temp_node;
        temp_node = temp_node -> n;
      }
    }
  }
}else
Printf ("Invalid position");
}
void display()
{
int count=0;
temp_node = first_node;
Printf ("Display linked list");
white (temp_node != 0)
  {
  printf("%d", temp_node -> a);
  Count++;
  temp_node = temp_node -> n;
  }
  printf ("No. of items %d", Count);
  }
```

```
int Count ( ){
int Count=0;
temp_node = first_node;
while (temp_node!=0)
{
  Count++;
  temp_node =temp_node ->next;
}
Printf ("No of Items in linked list %d", Count);
  return Count;
}
```

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} in the new list we should have {1, 4, 2, 5, 3, 6}

A:-

```
#include <stdio.h>
#include<stdlib.h>
{
 int value;
  struct Node* next;
};
Void Printlist (struct Node* head)
{
 Struct Node *ptr = head;
 while (ptr)
 {
  Printf ("empty \n");
 }
void push (struct Node** head, int value)
{
 struct Node* newNode = (struct Node*) malloc(size of
(struct Node));
```

```
newNode -> value = value;
newNode -> next = *head;
*head = newNode;
}
struct Node * shuffleMerge (struct Node* i,
struct Node *j)
{
 struct Node a;
 struct Node* tail = &a;
 a.next = empty;
while(1)
 {
  if(i==empty)
  {
   tail -> next = j;
   break;
  }
  else if(j==empty)
  {
  tail->next = i;
  break;
  }
  tail -> next = i;
   tail = i;
    i = i -> next.
```

```
tail -> next = j;
tail = j;
j = j->next;
}
}
return a.next;
}
int main(void)
{
  int keys[] = {1,2,3,4,5,6,7};
  int n = sizeof(keys)/sizeof(keys[0]);
struct node *i = empty, *j = empty;
for(int c=n-1; c>=0; c=c-2)
  push(&i, keys[c]);
for(int c = n-2; c>=0; c=c-2)
push(&j, keys[c]);
printf("first list");
printlist(i);
  printf("second list");
   printlist(b);
```

```
struct Node *head = shuffleMerge(a, b);
Printf("After merge");
 Printlist (head);
  return 0;
}
```

3) Find all the elements in the stack whose sum is equal to K.

```
#include<stdio.h>
int high = -1;
int a;
 Char stack[100];
void push (int c)
Char pop();
int  main( )
{
int i,t,e,f,g,h,K, count=0, sum=1;
Printf("Enter the elements");
 Scanf ("%d", & e);
for (i=0;i<e;i++)
{
 Print f ("Enter element");
 Scanf("%d", &F);
Push(f);
}
```

```
Printf ("Enter the sum");
Scanf("%d", &h);
for (d=0; d<e; d++)
{
g = Pop()
count+=t;
sum+=1;
if (count == h)
{
for (int j<0; j<sum; j++)
printf("%d", stack[j]);
K=1;
break;
}
push(g);
}
if(K!=1)
  Printf(" The elements do not add in the stack");
}
void push (int (a)
```

```
if (High==9a)
{
 Printf("stackfull");
 return;
}
 high = high+1;
 stack[high] = a;
}
char pop()
{
if (stack[high]==-1)
{
 Printf("empty");
 return 0;
}
 a= Stack[high];
 high = high-1;
      return a;
}
```

4) write a program to print the elements in a queue.

i) In reverse order

(ii) In alternate order.

```
#include<stdio.h>
#define SIZE 5
void Insert(int);
void delete();
int queue[5], front=-1, rear=-1;
void main()
{
int value, choice;
while(1)
  printf(" 1)Insertion\n2) Deletion\n 3) reverse\n4)
      Alternate\n 5) Exit");
  printf("Enter any value");
   scanf("%d", &choice);
   switch(choice){
case 1: printf("Enter the value");
 Scanf("%d", &value)
```

```
Case 2: delete();
break;
Case 3:
      printf ("Reversed:");
    for(int i=size;i>=0;i++
{
      if (queue[i]==0)
      Continue;
      printf("%d", queue[i]);
}
  break;
Case 4: printf("Alternative");
    for(int i=0; i<SIZE;i+=2)
  if (queue[i]==0)
    Continue;
     Printf ("%d", queue[i]);
}
    break.
case5: exit(0);
  default: printf ("wrong");
  }
}}
```

```
void insert (int value)
{
if (front == 0 && rear == SIZE - 1 || front == rear+1)
printf ("Queue is full");
    else
    {
     if (front == -1)
front = 0;
  rear = (rear+1) % SIZE;
  queue[rear] = value;
    Printf ("Insertion");
}}
void delete (){
 if (front == -1)
Printf ("Empty");
 else {
     printf ("Deleted %d", queue[front]);
      front = (front+1) % SIZE
      if (front == rear)
  front = rear = -1;
}}
```

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

(ii) write a program to add the first element of on list to another list.

Sol:- (i) Arrays are index based data structure where each element associated with an index or reference to the previous and next element.

(ii)

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
  int a;
  struct node* next;
};
void push(struct node** head_ref, int new_a)
{
struct node* newnode=(struct node*) malloc
 (sizeof(struct node));
new_node->a = new_a;
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->a);
   temp = temp -> next;
   }

   Printf (" \n");
}
```