Expt. No : 1 Date : Page No :

Expt. Name : Write a program to insert and delete and Element at Element at the n^{th} and k^{th} pointer in a linked list where n and k are taken from the user

A)

```
# 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;
    free (temp);
      return;
    }
      Node * temp = head;
```

Signature .....

```
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", &n);
   Scanf ("%d", &x);
   Insert (x, n);
   printf ("Enter the position to delete");
    Scanf ("%d", &k);
    Delete (k);
    print (x)
     return;
     }
```

Signature .....

Expt. No : 9 Date : Page No :

Name :

```
#include <stdio.h>
#include<stdlib.h>
Struct node{
   int data;
    Struct node next;
}
  Void print list (Struct node *head)
{
  pumtf ("%d→", (ptr→data));
     ptr=ptr→next;
     pumtf ("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 temp;
   Struct node *tail = false;
    false.next=Null;
   while(1)
   if(a==Null)
    {
```

Signature .....

```
    tail->next=b;
    break;
}
else if(b==NULL)
{
    tail->next=a;
    break;
}
else
{
    tail->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-1; i>0; i=i-2)
    push(&a, keys[i]);
for (int i=n-2; i>=0; i=i-2)
    push(&b, key[i]);
struct node * head = merge(a,b);
printlist(head);
}
```

Signature .....

Expt. No : 3 Date : Page No :

Expt. Name :

```
#include<stdio.h>
int top=-1;
int x;
char stack[100];
void push(int x);
char pop()
int main()
{
int 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==10){
for(int j=0;j<count;j++)
printf("%d", stack[j]);
   f=1
   break
}
```

Signature .....

Expt. Name :

```
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]=x;
}
  char pop()
{
 if (Stack[top]==-1)
{
   printf("\n Stack empty")
return 0;
}
  x=Stack[top];
   top=top-1
}
```

Signature .....

Expt. No : 4 Date : Page No :

Name :

```
#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***MENU***\n");
        printf("1.Insertion\n2.Deletion\n3.Reverse\4.Alternate");
        printf("\n\nEnter your choice");
        scanf("%d", &choice);
        switch(choice){
case 1: printf("Enter the value to be insert:");
        scanf("%d", &value);
        insert(value);
        break;
case 2: 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;
```

Signature .....

Expt. No :

t. Name :

```
Case 4:
    printf ("Alternate elements of queue)
    for (int i=0; i<SIZE; i+=2)
{
    if (queue[i]==0)
    Continue;
    printf("%d", queue[i]);
}
    break;

Case 5: exit(0)
default: printf("Wrong Selection")
    }
}
Void insert(int value){
    if((f==0 && r==SIZE-1) || f==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 delete(){
    if(f==-1)
    printf("\n Queue is empty")
```

Signature .....

Expt. No : Date : Page No :

Name :

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

Expt. No : 5 (i) Date : Page No :

Name :

How array is different from the linked list?

The major difference b/w 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 reference to the previous and next element

Expt. No : 8(ii) Date : Page No :

Name :

```
#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 printlist (Struct node *head)
{
  Struct node *temp = head;
   while (temp! = NULL)
    {
        printf ("%d", temp=>data);
        temp = temp -> next;
    }
    print ("\n");
}
```

Signature .....