Expt No : 1

Write a program to insert and delete and element at element at the n^{th} and k^{th} position 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 = 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;
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 ;
    print f ("Enter the position for inserting");
   scanf ("%. d", & n) ;
   scanf ("%. d", & 9 ) ;
   insert (x, n) ;
 print f ("Enter the position to delete ");
    scanf ("%. d", & u) ;
    Delete (u) ;
    print (x)
     return ;
    }
```

3 Expt. no: 2

```
# include < stdio.h >
#  include < stdoib.h >
  Struct node {
        int data;
        Struct node next;
      }
      void print list (struct node * head)
    {
      print f ("%d →", (ptr → data));
        ptr = ptr → next;
        print f ("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 face;
    struct node * tail = face;
      face.next = Null;
       while (1) {
       if (a == = Null)
        {
```

```
        tail->next = b;
        break;
    }
    else if (b==NULL)
    {
        tail->next = a;
        break;
    }
    else
    {
        tail->next = a;
        tail = a;
        a = a->next;
        tail->next = b;
    }
}
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[j]);
struct node *head = merge (a,b);
    print list (head);
}
```

Exp No: 3

```
#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 no. of elements within 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 == k){
    for(int j = 0; j < count; j++)
    printf("%d", stack[j]);
        f = 1
        break
}
```

```
push(i);
}
if (i!=1)
printf(" The elements in the stack don't
        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]==-1);
   {
    printf("\n stack empty")
  return 0;
  }
   x = stack[top];
     top = top-1
   }
```

Exp. no: 4

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

```
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 (if == 0 && r == size-1) || f ==
    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");
```

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

classmate
Date
Page

Exp.no: 5 (i)

How array is different from the linked list?
The major difference b/w array and linked list
regrads to their structure. Arrays are index ba
data structure where each element associated
with an index. on the other hand, linked
list relies on the other hand, linked list re
on reference to the previous and next ele

Exp. no : 5 (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 structure)
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;
 }
    printf ("\n");
}
```