1) Write a Program to insert and delete an element at the nth and k^{th} position in a linked list where n & k is taken from user

D:\Satya\Main\Sycom
CSE-F
AP19110010431

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
struct node *next;
};
struct node *cur, *temp;
void inpos(struct node*);
void delpos(struct node*);
void main(void)
{
struct node *s;
int ch;
s=NULL;
do
{
printf("\n1. Inpos\n");
printf("2. Delpos\n");
printf("3. Exit\n");
printf("Enter choice");
scanf("%d",&ch);
        switch(ch)
        {
        case 1 :inpos(s);
                break;
        case2: delpos(s);
```

```
}switch(ch!=3)
}
void inpos(struct node *x)
{
int pos, c=1;
curr = x;
printf("Enter the pos to be inserted:")
scanf("%od", &pos);
        while(curr->next != NULL)
        {
        c++;
          if(c==pos)
        {
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter the number: ");
        scanf("%od", &temp->n);
        temp->next = curr->next;
        curr->next = temp;
        break;
        }
   }
}
void delpos(struct node * x)
{
int pos, c=2;
curr = x;
printf("Enter the pos to be deleted:");
scanf("%od", &pos);
        while(curr->next!=NULL)
        {
        c++;
                if (c==pos)
                {
                temp = curr->next;
                curr->next = curr->next->next;
                free(temp);
                }
        curr = curr->next;
        }
```

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

Sol:

```
#include<stdio.h>
#include<stdlib.h>
#include<assert.h>
struct node.
{
    int data;
    struct node *next;
};
void Movenode (struct node **x; struct node **y);
struct node* SortedMerge (struct node *a, struct node *b).
{
    struct node dummy;
    struct node *tail = &dummy;
    dummy.next = NULL;
    while (1)
    {
        if (a == NULL)
        {
            tail->next = b;
            break;
        }
        else if (b == NULL)
        {
            tail -> next = a;
```

```
}
if (a->data <= b->data).
{
    MoveNode(&(tail->next), &a);
}
else
{
    MoveNode(&(tail->next), &b);
}
    tail = tail->next
}
return (dummy.next);
}
void push(struct node ** head_ref, int new_data)
{
    struct node *new_node = (struct node*) malloc (sizeof (struct node));
        new_node->data = new_data;
        new_node->next = (*head_ref);
        (*head_ref) = new_node;
}
void printList (struct node *node)
{
    while(node != NULL)
    { printf ("%d", node->data);
      node = node->next;
    }
}
```

```
int main()
{
    struct node* res = NULL;
    struct node* a = NULL;
    struct node* b = NULL;
    push(&a, 1);
    push(&a, 2);
    push(&a, 3);
    push(&b, 4);
    push(&b, 5);
    push(&b, 6);
    res = sortedMerge(a, b);
    printf("Merged linked list is : \n");
    printList(res);
    return 0;
}
```

Output:

Merge linked list is

4 2 5 3 6.

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

```
#include<stdio.h>
#define size 5
int stack[size], top = -1;
void push(int);
int pop();
int isempty();
int isfull();
void traverse(int);
int main()
{ int choice, ele, k;
  printf("enter k value:");
  scanf("%d", &k);
  switch(choice)
  {
  case 1:
    printf("enter the element:");
    scanf("%d", &ele);
    push(ele);
    break;
  case 2:
  ele = pop();
  printf("%d", ele);
  break;
  case 3:
  traverse(k);
```

```
        break;
    case 4:
        break;
    default:
        printf("invalid input");
    }
  }
}
int isfull()
{ if(top == size-1)
  {
    return 1;
  }
  else
  {
    return 0;
  }
}
int isempty()
{ if(top == -1)
  {
    return 1;
  }
  else
  {
    return 0;
  }
}
void push(int ele)
{
  if(isfull())
  {
    printf("stack is filled");
  }
  else
  {
    top++;
    stack[top] = ele;
  }
}
int pop()
{ if(isempty())
  { printf("stack is empty");
  }
  else
  {
    return stack[top];
    top--;
  }
}
void traverse(int k)
{
  if(isempty())
  {
    printf("stack is empty");
  }
  else
  {
```

```
printf("single elements are\n");
for(i=0; i<top; i++)
{
if(stack[i]==k)
{
printf("pairs are\n");
for(i=0;i<top;i++)
{
if(stack[i]==k)
{
printf("%d\n",stack[i]);
}
}
printf("pairs are\n");
for(i=0;i<top;i++)
{
for(j=i+1;j<top;j++)
{
temp=stack[i]+stack[j];
if(temp==k)
{
printf("%d %d\n",stack[i],stack[j]);
break;
}
}
}
}
}
```

9) Write a program to print the elements in a queue

i. in reverse order

ii. in alternate order

Ans:

```
#include<stdio.h>
#include<stdlib.h>
#define size 5
int queue[size];
int front=-1;
int rear=-1;
void insert(int);
int remove();
int traverse();

int main()
{
int choice,ele;
while(1)
{
printf("1.add 2.delete 3.traverse 4.exit \n");
printf("Enter the Choice:");
scanf("%d",&choice);
switch(choice)
{
case 1;
printf("enter the element:");
scanf("%d",&ele);
insert(ele);
break;
```

```
            case 2:
             remove ();
             break;
            case 3:
             traverse ();
             break;
            case 4:
             exit (0);
        }
    }
}
void insert (int ele)
{
 if (rear == size-1)
 {
 printf ("queue is filled");
 }
 else
 {if (front == -1)
 {
 front = 0;
 }
  rear ++;
  queue[rear] = ele;
 }
}
int remove ()
{
 if (front == rear)
```

```
{
  printf ("queue is empty");
}
else
{
  print ("%d", queue[front]);
  front++;
  if (front == rear)
  {
    front = rear = -1;
  }
}
}
int traverse ()
{
  if (front == rear)
  {
  printf("queue is empty");
}
else
{
int i;
printf("the reverse order is \n");
for (i=rear; i>0; i--)
{
  printf("%d \t", queue[i]);
}
printf ("In the alternative order is \n");
for (i=0; i<=rear, i+=2)
{
  printf("%d \t", queue[i]);
}
printf("\n");
}
```

5.

(i) How array is different from the linked list

Ans: The major difference between array and linked list due to their structure. Arrays are index based data structure where each element associated with a index. On the other hand linked list relies on references where each node consists of data and the refrences to the previous and next element.

(ii) write a program to add the 1st 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 list 1 and {5,6} for list 2

Ans

```
#include <stdio.h>
#include <stdlib.h>
{ int data;
  struct Node* next;
};
{
  struct Node *ptr = head;
  while (ptr)
  {
    printf("%d -> ", ptr->data);
    ptr = ptr->next;
  }
```

```
    print ("NULL\n");
}
void push (struct Node** head, int data)
{
  struct Node* newNode = (struct Node*) malloc (sizeof (struct Node));
    newNode->data = data;
    newNode->next = *head;
    *head = newNode;
}
void MoveNode (struct Node** destRef, struct Node** SourceRef)
{
  if (*SourceRef == NULL)
    return;
  struct Node* newNode = *SourceRef;
  *SourceRef = (*sourceRef)->next;
  newNode->next = *destRef;
  *destRef = newNode;
}
int main (void)
int keys[] = {1,2,3};
int n = sizeof(keys) / sizeof(keys[0]);
 struct Node *a = NULL;
 for (int i = n-1; i >= 0; i--)
    push (&a, keys[i]);
```

```
struct Node* b = NULL;
for (int i = 9; i >= 0; i--)
push(&b, 2 * keys[i]);
MoveNode(&a, &b);
printf("First List : ");
printList(a);
printf("Second List : ");
printList(b);
return 0;
}
```