DATA STRUCTURES

ASSIGNMENT-4

CHANDANAM.TANMAYEE
CSE-F
AP19110010402

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.

```
#include<stdio.h>
#include<malloc.h>
#include <stdlib.h>
struct node {
    int value;
    struct node *next;
};
Void insert();
void display();
Void delete();
int Count();
typedef struct node DATA_NODE;
DATA_NODE *head_node,*first_node,*temp_node=0,*prev_node,
 next_node;
 int data;
 int main(){
   int option=0;
   printf("singly linked list Example-All operations \n");
   While(option<5){
     printf("\n Options \n");
     printf("1: Insert into Linked List \n");
     printf("2: Delete from Linked List \n");
     printf("3: Display Linked List \n");
     printf("4: Count Linked List \n");
printf ("Others : Exit () \n");
printf ("Enter your opinion: ");
Scanf ("%d", &option);
switch (option) {
  Case 1 :
    insert ();
    break ;
  case 2:
    delete ();
    break ;
  case 3:
    display ();
    break;
  Case 4:
    count ();
    break;
  default:
    break;
  }
}
return 0;
}
Void insert (){
  printf ("\n Enter Element for insert Linked list :\n");
  Scanf ("%d", &data);
  temp_node =(DATA_NODE *) malloc (sizeof (DATA_NODE));
  temp_node -> Value = data ;
  if (first_node = =0) {
      first_node =temp_node ;
  } ~~else{~~
    else{
      head_node ->next =temp_node ;
    }
    temp_node -> next = 0;
    head_node =temp_node ;
  }   fflush (stdin);
```

```
Void delete(){
    int CountValue ,pos ,i =0 ;
      CountValue = count();;
      temp_node = first_node;
      printf ("\n Display Linked list : \n");

      printf ("\n Enter Position for Delete Element : \n");
       scanf ("%d", &pos);
       if (pos>0 && pos <= CountValue) {
         if (pos = =1) {
           temp_node = temp_node -> next;

           first_node =temp_node;
           printf ("\n Deleted Successfully \n\n");
         }
         else
         {
           while (temp_node != 0) {
             if (i= = (pos-1)) {
              prev_node -> next = temp_node -> next;
              if (i = = (countvalue -1))
              {
                        head_node = prev_node;

                 }
              printf ("\n Deleted Successfully \n\n");
              break;
            }
             else {
               i++;
              prev_node =temp_node;
              temp_node =temp_node -> next;
            }
          }
        }
      }
      else
       printf("\n Invalid Position \n\n");
    }
```

```
void display(){
    int Caunt = 0;
    temp_node = first_node;
    printf("\n Display Linked List : \n");
    While (temp_node != 0){
        printf("# %d# ", temp_node -> value);
        count++;
        temp_node = temp_node -> next;
    }
    printf("\n No of items in Linked list : %d \n", count);
}
int count (){
    int count = 0;
    temp_node = first_node;
    While (temp_node != 0){
        count ++;
        temp_node = temp_node -> next;
    }
    printf ("\n No of items In Linked List: %d \n", count);
    return count;
}
```

Output:

```
1: Insert into Linked List
2: Delete from Linked list
3: Display linked list
4: Count Linked list
others: Exit ()
Enter your option: 1
Enter Element for Insert Linked list :
3
options
1: Insert into Linked list
2: Delete from Linked list.
```

```
3: Display Linked List
4: Count Linked List
others: Exit()
Enter your option: 3.
Display Linked list :
# 3 #
No of Items in Liked List :1
options :
1: Insert into Linked List
2: Delete from Linked List
3: Display Linked List
4: Count Linked List
others : Exit()
Enter your option: 6.
```

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 {1,4,2,5,3,6}

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

```
        printf ("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;
}
struct Node* shuffleMerge (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;
               break;
          }
          else
          {
              tail -> next = a;
               tail = a;
                a = a -> next;

              tail -> next = b;
               tail = b;
               b = b -> next;
          }
      }
      return dummy.next;
}
```

```
int main(void)
{
        int keys[] = {1,2,3,4,5,6,7};
        int n = sizeof(keys)/sizeof(keys[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, keys[i]);
          printf("First list : ");
          printList(a);
          printf("Second List : ");
          printList(b);
          struct Node* head = shuffleMerge(a,b);
          printf("After Merge: ");
          printList(head);

          return 0;

    }
```

OUTPUT :

```
First list : 1 -> 3 -> 5 -> 7 -> NULL
Second List : 2 -> 4 -> 6 -> NULL
After Merge : 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> NULL
```

3) Find all the elements in the stack whose sum is equal to K (where K is given from user).

```
#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 ~~number~~ sum of 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 (t);
    }
     if (f!=1)
     printf ("The elements in the stack dont 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] == -1)
   {
     printf ("\n Stack is EMPTY!!!\n");
     return 0;
   }
   x = Stack [top];
   top = top - 1;
     return x;
 }.
```

OUTPUT :

```
Enter the number of elements in the stack 4
 Enter next element 3
 Enter next element 1
 Enter next element 8
 Enter next element 9
 Enter the sum to be checked 21
 The elements in the stack dont add up the sum.
```

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

i. in reverse order

ii. in alternate order.

```
#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 ("1.Insertion \n 2.Deletion \n 3. Print Reverse \n 4.Print
                                   .Alterate \n 5 Exit");

   printf ("\n Enter 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;
           case 4 :
              printf ("Alternate elements of queue are: ");
              for( int i=0 ; i<SIZE ; i+=2)
      {
              if (queue [i]==0)
                 continue;
                printf ("%d", queue [i]);
        }
            break;
    Case 5: exit (0);
    default : printf ("\n Wrong Selection!!! Try again!!");
        }
    }}
```

```
Void insert (int value) {
    if(f==0 && r=SIZE -1)|| f==r+1)
       printf ("\n Queue is Full!! Insertion is not possible!!");
       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!! Deletion is not posible!! ");
    else {
      printf ("\n Deleted : %d", queue [f]);
       f= (f+1) % SIZE;
       if (f==r)
      f=r=-1;

 } }
```

OUTPUT :

```
 1.Insertion
 2. Deletion
 3. Print Reverse
 4. Print Alternate
  5. Exit
Enter your choice: 1
Enter the Value to be insert: 5

 Insertion Sucess!!!

1. Insertion
2. Deletion
3. Print Reverse
```

```
4. Print Alternate
5. Exit
Enter your choice:1
Enter the value to be insert :3

Insertion Success!!!

1. Insertion
2. Deletion
3. Print Reverse
4. Print Alternate
5. Exit
Enter your choice:3
The Reversed queue is: 3 5

1. Insertion
2. Deletion
3. Print Reverse
4. Print Alternate
5. Exit
Enter your choice:5
```

Program-5:

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

Sol: The major difference between array and linked list regards to their structure. Arrays are index based data structure, ~~Arrays~~ where each element associated with an index, while a linked list is a data stoucture which contains a sequence of the elements where each element is linked to its next element.

(ii) Write a program to add the first element of one list to another list for 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.

```
#include<stdio.h
#include <stdlib.h>
struct Node
{
    int data;
    struct Node* next;
};
void printList (struct Node* head)
{
    struct Node* ptr = head;
    while (ptr)
    {
        printf ("%d"->", ptr->data);
        ptr = ptr ->next;
    }
    printf ("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 ; l>=0 ; i--)
          push (&a ,key[i]);

      Struct Node* b = NULL
     for(int i =0; i<n; i++)
          push (&b, 2* keys [i]);

          MoveNode (&a,&b);

          printf (" First List : ");
         printList (a);
         printf ("Second List :  ");
          printList (b);

          return 0;

}
```

OUTPUT:

```
First List : 6 ->1 ->2 ->3 -> NULL
Second List : 4 ->2 -> NULL
```