Assignment - 4

A Pavan
AP19110010469
CSE - F

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 ();

type def 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 option: ");
    Scanf ("%d", &option);
    Switch (option) {
      case 1:
        insert();
         break;
      case 2:
        delete();
        break;
      case 3;
        display();
        break;
      case 4:
        count();
        break;
   }
}
 return 0;
}
 Void insert(){
Printf ("\n Enter Element for Insert linked List : \n");
  Scanf ("%d", &data);
temp_node = (DATA_NODE*) malloc(size of (DATA_NODE));
  temp_node -> value = data;
```

```
    if (first_node == 0) {
      first_node = temp_node;
    } else {
      head_node->next = temp_node;
    }
  temp_node->next = 0;
  head_node = temp_node;
  flush(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 == (count value - 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 count=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:-

Menu

1) Create
2) Display
3) Insert a node at specified position
4) Delete node
5) Exist

Enter your choice: 1
Enter the data value for the node: 20

Menu

1. Create
2. Display
3. Insert a node at specified position
4. Delete node
5. Exist

Enter your choice: 1
Enter the data value for the node: 50

Menu

1. Create
2. Display
3. Insert a node at specified position
4. Delete node
5. Exist

Enter your choice: 3
Enter the position for the new node to be inserted
Enter the data value of the node: 44

Menu

1. Create
2. Display
3. Insert at specified Position
4. Delete from specified Position
5. Exit.

Enter your choice : 4

Enter the position of the node to be deleted : 3

Position not found.

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}

Program:-

```
#include <stdio.h>
#include <stdlib.h>

// Data structure to store a linked list node
struct node
{
    int data;
    struct Node * next;
};

// Helper function to print given linked list
void printList(struct Node * head)
{
    struct Node * ptr = head;
    while(ptr)
    {
        Print("%d-> ", ptr->data);
        ptr = ptr -> next;
    }
    Printf("NULL\n");
}

// Helper function to insert new Node in the beginning
of the linked list void Push(struct node** head, int data)
{
    struct Node * newNode = (structNode*) malloc(sizeof(
                                                structnode));
```

```
    new Node -> data = data;
    New node -> next = *head;
     *head = new Node;
// Function to construct a linked list by merging
   alternate nodes of
// two given linked lists using dummy node
Struct node * Shuffle Merge (struct Node *a, struct Node *b)
 {
    Struct Node dummy;
    Struct Node * tail= &dummy;
    dummy.next = NULL;
    while(1)
    {
       // empty list cases
        if (a==NULL)
        {
          tail->next=b;
           break;
        }
        else if (b == NULL)
         {
           tail ->next=a;
            break;
         }
       // common case: move two nodes to tail
        else
        {
          tail ->next=a;
           tail = a;
            a = a->next;
```

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

// main method
int main(void)
{
    // input keys
    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]);
    // Print both linked list
    Printf("First list: ");
    PrintList(a);
    Printf("second List: ");
    PrintList(b);
    struct Node *head = ShuffleMerge(a,b);
    Printf("After Merge: ");
    PrintList(head);
    return(0);
}
```

Input - output:

First List: 1 → 3 → 5 → 7 → NULL

Second List: 2 → 4 - 6 → NULL

After merge: 1 → 2 → 3 → 4 → 5 → 6 → 7

}

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

```
#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 == 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)
{
  Print f ("In stack is full !!! In");
return }
}
top = top+1;
 stack[top] = x;
}
char Pop()
{
  if (stack[top] == -1);
 {
  Printf ("In stack EMPTY !!! In");
  return 0;
 }
  x = stack[top];
```

```
    top = top - 1;
    return x;
}
```

Input - output :-

```
Enter number of elements in stack 3
Enter element 4
Enter element 5
Enter element 4
Enter the sum to be checked 30
The elements in the stack do not equal to sum.
```

4) Write a program to print the elements in queue

i) In reverse order

ii) In Alternate order.

Program:-

```
#include<stdio.h>
#define SIZE 10
Void insert(int);
Void delete();
int queue,choice
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\n 3.Print Reverse\n
                    4.Print Alternate\n 5.Exit\n");
  Printf("\n Enter your choice.");
  switch(choice){
case 1: Printf("Enter the value of 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 the 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("In wrong selection !!! Try again !!!");
   }
}}
void insert(int value){
if ((f == 0 && r == size -1) || f == r+1)
 Printf (" In 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 deletion(){
 if (f == -1)
 Printf("\n queue is Empty !!! Deletion is not Possible");
 else{
  Printf("\n Deleted : %d", queue[f]);
   f = (f+1) % SIZE;
   if (f==r)
 f = r = -1
}}
```

Menu

1. Insertion
2. Deletion
3. Print Reverse
4. Print Alternate
5. Exit

Enter your choice: 1

Enter the value to insert: 6

Insertion success !!!

MENU

1. Insertion
2. Deletion
3. Print Reverse
4. Print Alternate
5. Exit

Enter your choice: 1

Enter the value to be insert: 20

Insertion success !!!

MENU

1. Insertion
2. Deletion
3. Print Reverse
4. Print Alternate
5. Exit

check your choice: 3

The reversed queue is : 20 6

MENU

1. Insertion
2. Deletion
3. Print Reverse
4. Print Alternate
5. Exit

Enter your choice : 4

Alternate elements of queue are : 6

MENU

1. Insertion
2. Deletion
3. Print Reverse
4. Print Alternate
5. Exit

Enter your choice : 5

5)

i) How array is different from linked list

Difference between Array and linked list The major difference between Array and Linked list regards to their structure Arrays are index based data structure where each element assiociated with an index. on the other hand, Linked list relies on references where each node consists of the data and the refesences to the previous and next element.

ii) write a program to add the first element of one list for example we have {1, 2, 3} in list 1 and {4, 5, 6} in List 2 we have get {4, 1, 2, 3} as output for List 1 and {5, 6} for List 2.

Program:-

```
#include<stdio.h>
#include<stdlib.h>

// Data Structure to store a linked list node
Struct Node
{
  int data;
  Struct Node * Next;
};

// Helper function to Print given linked list
Void PrintList (Struct Node *head)
   {
     Struct Node * Ptr = head;
      while (Ptr)
```

```
    {
      Printf ("%d -> ", ptr->data);
       ptr = ptr -> next;
    }
    Printf("NULL \n");
}

// Helper function to insert new Node in the beginning
  of the linked list
  Void Push (struct Node ** head, int data)
{
   StructNode * newNode = (struct Node*) malloc (size of
                                                 (struct Node));
      new Node -> data = data;
      new Node -> next = *head;
       *head = newNode;
  }

// Function take the Node from the front of the source,
  and move it
// to the front of destination
Void Move Node (Struct Node ** destRef, StructNode **
                                                 SourceRef).
{
  // if the source list empty, do nothing
    if (* Source Ref == NULL)
        return;
```

```
Struct node * newnode = * sourceRef; // the front source
                                        Node

*SourceRef = (*SourceRef)->next; //Advance the source
                                  Pointer.

newNod ->next = *desRef; // Link the old dest off the
                            New node.

  *destRef = newNode; // Move dest to point to the
                         New node.

}

// main method
int main (void)
{
   // input keys
   int keys [] = {1,2,3};
   int n = size of (keys) / size of (keys[0]);

   // construct first linked list
    struct Node * a = NULL;
    for (int i = 0; i<n; i--)
        Push (&a ; keys[i]);

   // construct second Linked list
     Struct Node *b = NULL;
     for (int i=0; i<n; i++)
         Push (&b; 2* key[i]);

// move front node of the b, and move it to the front
   of the a
  Move Node (&a, &b);
```

```
// Print both lists
printf("First List: ");
printList(a);
printf("second List: ");
printList(b);
return 0;
}
```

Input - output :

First List : $6 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow$ NULL

second List : $4 \rightarrow 2 \rightarrow$ NULL