B. Haritha DSA ASSIGNMENT-4 AP19110010287 CSE-G

1. Write a program to insert and delete an element at nth and kth position in the linked list where n and k is taken from the user.

Code:

```
# include <stdio.h>
# include <stdlib.h>
# define initMemory()  ((struct node *)malloc(sizeof(struct node)))
struct node
{
    int data;
    struct node *next;
};
// function for creating a linked list
struct node * create list ()
{
    int i, n;
    struct node *p, * Head = Null

// inputting n
// n is the number of elements we want the linked list to be
        created with

Printf ("\n How many elements to enter ? ");
scanf (" %d", &n);

if (n == 0)
{
    return Head;
}
```

```
for (i=0 ; i<n ; i++)
{
    if (i==0)
    {
      Head = initmemory ( );
       p = Head ;
    }
  else .
  {
    // initializing current ptr's next to the address of newptr
       p -> next = init memory ( );
       p = p -> next ;
  }
    Printf (" Enter the %d th element ", i);
    scanf (" %d ", & p -> data);
}.
// setting the last node's next as null.
   p -> next = NULL ;
   return ( Head );
}
// Function which displays a linked list
void display ( struct node * head )
{
    if (head == NULL)
    { printf (" no element in the list ");
    } return;
```

```
Struct node * ptr = head ;
While ( Ptr != NULL )
{
    Printf (" %d ", ptr -> data) ;
    Ptr = ptr -> next ;
} }

// function inserts a node at position n
Void insert AtN (struct node ** head, int n)
{
    int i ;
    Struct node * newptr , * ptr , * tmp ;
    newptr = init memory ( ) ;
    Printf ("Enter the element to be inserted ");
    scanf (" %d ", & newptr -> data) ;
/* if n is 0, first element head shifts to new node */
    if (n == 0)
    {
        newptr -> next = * head ;
        * head = newptr ;
        // exiting the function.
        return ;
    }
    ptr = * head ;
    // loop which stops before n-1 element
```

```
for (i=0 ; i<n-1 ; i++)
{
    if ( ptr -> next == NULL)
    {
        if ( i! = n-1 )
        {
            Printf(" list is not initialized till n \n entering the newnode at
                                                    end \n");
        }
        Ptr -> next = new ptr ;
        newptr -> next = NULL ;
        return ;
    }
    else
    {
        ptr = ptr -> next ;
    }
}
    tmp = ptr -> next;
    ptr -> next = newptr ;
    newptr -> next = tmp ;
}
// function deletes a node at position k
Void deleteAtk (struct . node ** head , int k)
{
    int i ;
    Struct node * temp ;
    if ( k == 0 )
    { Printf ("\n%d is deleted \n", (*head) -> data);
```

```
    tmp = *head;
    *head = (*head) -> next;
    free (tmp);
    return;
}
Struct node *ptr = *head;
for (i=0; i<k-1; i++)
{
    if (ptr -> next == NULL)
    {
        printf ("\n no element in position %d \n", k);
        return;
    }
    else
    {
        ptr = ptr -> next;
    }
}
tmp = ptr -> next;
Printf ("%d is deleted \n", tmp -> data);
ptr -> next = tmp -> next;
free (tmp);
}
// main
int main()
{
    int n, k
```

```
Struct node * head = createList ();
Printf ("\n enter n, index of where you want to add a node");
Scanf ("%d", &n);
insertAtN (&head, n);
Printf ("\n after entering, list looks like \n");
display (head);
Printf ("\n enter k, index of where you want to delete a node \n");
Scanf ("%d", &k);
deleteAtK (&head, k);
Printf ("\n after deleting, list looks like \n");
display (head);
return 0;
}
```

Output :

How many elements to enter ? 4
Enter the 0th element 6
Enter the 1th element 7
Enter the 2th element 8
Enter the 3th element 9

Enter n, index of where you want to add a node 2
enter the element to be inserted 3
after entering, list looks like.
6 7 3 8 9
Enter k, index of where you want to delete a node
7
7 is deleted.
after deleting, list looks like
6 3 8 9

2. Construct a new Linked List by merging alternate nodes of 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}.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#define initMemory() ((struct node *) malloc (sizeof(struct node)))
struct node
{
    int data;
    struct node * next;
};
// function for creating a linked list
struct node * createList()
{
  int i, n;
  struct node *p, * Head = NULL;
  Printf("\n How many elements to enter ?");
  Scanf("%d", &n);
  if (n==0)
  {
     return Head;
  }
  // entering the elements
 for (i=0; i<n; i++)
 { if (i==0)
    { Head = initMemory();
```

```
        p = Head ;
    }
    else
    {
        p->next = initMemory ( );
        p = p->next ;
    }
    Printf (" Enter the %d th element ", i );
    Scanf ( " %d ", &p->data );
}
    p -> next = NULL ;
    return ( Head );
}
// function which displays a linked list
Void display (struct node * head )
{
    if ( head == NULL)
    { printf (" no element in the list ");
      return ;
    }
    Struct node * ptr = head ;
    while ( ptr != NULL)
    {
        printf( " %d ", ptr -> data);
        ptr = ptr->next ;
    }
}
```

```
// function to join two lists alternatively.
Struct node * JoinAlt (Struct node * list1, Struct node * list2)
{
   Struct node * newlist = NULL, * tmp;
   if (list1 != NULL)
   {
      newlist = list1;
      tmp = list1;
      list1 = list1 -> next;
   }
   // when list 1 is empty and list 2 is not
   else if (list2 != NULL)
   {
      newlist = list2;
      return
   }
    else
    {
      return newlist;
    }
    int i = 0;
    while (1)
    {
      if (i%2 != 0 && list1 != NULL)
      {
         tmp -> next = list1;
         list1 = list1 -> next;
      }
```

```
else if (list2 != NULL)
{
    tmp -> next = list2;
    list2 = list2 -> next;
}
else
{
    break;
}
tmp = tmp -> next;
itt;
}
return newlist;
}
int main()
{
    struct node *list1 = createlist(), *list2 = createlist();
    printf("\n list1 was \n");
    display(list1);
    printf("\n list2 was \n");
    display(list2);

    // merging list
    struct node *newlist = joinAlt(list1, list2);
    printf("\n after merging lists: ");
    display(newlist);
    return 0;
}
```

Output

How many elements to enter ? 3
Enter the 0th element 1
Enter the 1th element 2
Enter the 2th element 3
How many elements to enter ? 3
Enter the 0th element 4
Enter the 1th element 5
Enter the 2th element 6
List 1 was
1 2 3
List 2 was
4 5 6
after merging lists : 1 4 2 5 3 6

(3) Find all the elements in the stack whose sum is equal to k (where k is the 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");
   Scant ("%d", &a);
   Push (a);
}
Printf ("Enter the sum to be checked");
Scanf ("%d", &k);
for (i=0; i<n; i++)
{
   t = pop();
   Sumt = t;
   countt++ = 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 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] == -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 3
Enter next element 2
Enter next element 3
Enter next element 5
Enter the sum to be checked 10.
2 3 5
```

(4) Write a program to print the elements in a queue
   i. in reverse order
   ii. in alternative order

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node{
    int data;
    Struct Node * Next;
} node;
typedef struct Queue{
    node * front, * rear;
} queue;
node * NewNode (int k){
  node * temp = (node *) malloc (size of (node));
   temp -> data = k;
   temp -> next = NULL;
```

```
return temp;
}
queue createQueue() {
    queue q;
    q.front = q.rear = NULL
    return q;
}
Void enqueue (queue *q, int k) {
    node * temp = newNode(k);
    if (q -> rear == NULL)
    {
        q -> front = q -> rear = temp;
        return;
    }
    q -> rear -> next = temp;
    q -> rear = temp;
}
Void displayAlt (queue q) {
    while (q.front != NULL) {
    printf(" %d -> ", q.front -> data);
    if (q.front -> next != NULL) q.front = q.front -> next -> next;
    else break;
}
printf("NULL \n");
}
```

```
Void display Rev (queue q ) {
    If (q.front == NULL) {
        Printf ("NOLL ");
        return;
    }
    int temp = q.front -> data ;
    q. front = q. front -> next ;
    display Rev (q);
    Printf (" Reverse <- %d ", temp);
}

int main()
{
    queue q = create Queue ();
    int n, num ;
    Printf ("Enter the number of element you want in the queue \n");
    Scanf ("%d ", &n);
    while (n--) {
    Printf ("number \n");
    scanf ("%d", & num);
    enQueue (eq , num);
}
    display Rev (q);
    Printf ("\n ");
    display Alt (q);
    return 0;
}
```

Output:

Enter the number of element you want in queue

5

number

4

number

5

number

6

number

7

number

8

NULL ← 8 ← 7 ← 6 ← 5 ← 4 ← 8

4 → → 6 → → 8 → → NULL

(5) is how array is different form linked list

Arrays	Linked list
Fixed size : Resizing is expensive	Dynamic size.
Insertions and deletions are inefficient : Elements are usually shifted.	Insertions and Deletions are efficient : No shifting
Random access i.e., efficient indexing	No random access → Not suitable for operations requiring accessing elements by index such as sorting

No memory waste if the array is full or almost full; Otherwise may result in much memory waste

Since memory is allocated dynamically (acc. to our need) there is no waste for memory.

Sequential access is faster
[Reason: Elements in contiguous memory locations]

Sequential access is slow
[Reason: elements not in contiguous memory locations]

(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>
#define initmemory() ((struct node *)malloc(sizeof(struct node)))
Struct node
{
   int data;
   struct node *next;
};
Struct node* create list()
{
   int i, n;
   Struct node *p, *head = NULL;
   Printf("\n How many elements you want to enter?");
    Scanf("%d", &n);
```

```
if (n == 0)
{
return head;
}
for (i=0; i<n; i++)
{
  if (i == 0)
  {
    Head = init memory ();
    p = Head;
  }
  else
  {
    p -> next = init memory ();
    p = p -> next;
  }
  printf ("Enter the %d th element ", i);
  scanf ("%d", &p -> data);
}
p -> next = NULL;
return (Head);
}
Void display ( Struct node * head )
{
  if (head == NULL)
  {
    printf ("no element in the List");
    return;
  }
  Struct node * ptr = head;
```

```
while (ptr != NULL)
{
    printf("%d ", ptr -> data);
    ptr = ptr -> next;
}
}

void shift first element (struct node ** list1, struct node **list2)
{
    struct node * tmp = * list2;
    * list2 = tmp -> next;
    tmp -> next = * list1;
    * list1 = tmp;
}

int main()
{
    struct node * list1 = createlist(), * list2 = create list();
    shift first element (& list1, & list2);
    printf("\n list 1 is: ");
    display (list1);
    printf("\n list 2 is");
    display (list2);
}
```

Output

How many elements you want to enter? 3
Enter the 0th element 11
Enter the 1st element 12
Enter the 2th element 13
How many elements to enter? 3
Enter the 0th element 14
Enter the 1th element 15
Enter the 2th element 16
List1 is : 14 11 12 13
List2 is : 15 16