E. Rohith
AP19110010282
CSE H

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.

Program :-

```
#include < stdio.h >
# include < stdlib.h >
Void insert ( Node *, int, int )
int six = 0 ;
Struct node {
int data ;
Struct node * next ;
}
Node * get_node (int data)
{
 Node * newnode = (Struct node * )malloc(newnode );
      newnode -> data = data ;
      newnode -> next = NULL ;
      return newnode ;
}
```

AP19110010282

```
Void insert (node * (current, int position, int data)
{
 if (Pos <-1 || Pos > size+1)
    printf (" Invalid " );
 else {
    while (pos--)
 {
  If (Pos == 0)
 {
   Node* temp = getnode(data);
  temp -> next = *Current ;
   *Current = temp ;
  }
 else {
  Current = & (*Current) -> next ;
}
 Size++ ;
}
}
Void print ( Struct node *head )
{
  while (head != NULL)
  {
    Printf (" %d", head -> data);
    head = head -> next ;
  }
  Printf ("\n");
 }
 Void del (struct node * head_ref , int position)
 {
    if (head_ref = NULL)
    return ;
```

```
temp = head_ref ;
if (pos == 0)
{
  * head_ref = temp -> next ;
   free (temp) ;
   return ;
for (int i=0; temp != NULL && i < pos-1 ; i++)
temp = temp -> next ;
free (temp -> next) ;
temp -> next = next ;
}
int main ()
{
 struct node* head = NULL ;
 Push (&head, 12) ;
 Push (& head, 15);
 Push (& head, 8);
 insert (&head, 4, 9, 3);
 delete (&head, 12);
 return 0 ;
}
```

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}.

Step 1 :- statement of the program

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}.

Step 2 :- Explanation of the program

Here first we should Create two new Linked Lists then we should merge alternate nodes of second Linked List with first Linked List.

Step 3 :- Steps and Algorithm involved in the program.

1) Create a Structure.
2) Function to insert a node at beginning
3) Function to print singly Linked List
4) Function that inserts nodes of Linked q into P alternate position.
5) Program to test the above function.

Step 4 :- Code in C language

```
// Program to merge a Linked list of alternate nodes.
#include <stdio.h>
#include <stdlib.h>
Struct Node {
    int data;
    struct Node *next;
};
// Function to insert a node at beginning
Void Push (struct Node **head_ref, int new_data)
{
 StructNode *new node = (StructNode*) malloc (Size of (StructNode));
 new_node -> data = new_data;
new_node -> next = (*head_ref);
(*head_ref) = new_node;
}
// Function to print the singly Linked list
void Print list (struct Node *head)
{
 Struct Node *temp = head;
 while (temp != NULL)
 {
   Printf ("%d", temp -> data);
   temp = temp -> next;
 }
 Printf ("\n");
}
```

```
// Function that inserts nodes of Linked List q into p at alternate
   positions
Void merge (struct Node *a, struct Node **v)
{
    struct Node *a_first = a, *v_first = *v);
    Strud Node *a_next, *v_next ;
    while (a_first != NULL && v_forst != NULL)
    {
      a_next = a_first -> next ;
      v_next = v_first -> next ;
      v_first -> next = a_next ;
      a_first -> next = v_first ;
      a_first = a_next ;
      v_first = v_next ;
    }
     *q = v_first ;
}

// Program to test above functions
int main ( )
{
  struct Node *p = NULL, *q = NULL ;
  Push (&a, 3) ;
  Push (&a, 2);
  Push (&a, 1) ;
  Printf ("First Linked List a : \n") ;
  PrintList (a);
  Push(&v, 8) ;
  Push (&v, 7) ;
  Push (&v, 6) ;
  push (&v, 5) ;
  Push (&v, 4) ;
```

```
Printf ("Second linked list v: \n");
Printlist (v)
merge (a, &v);
Printf ("Merged first linked list a: \n");
Printlist (a);
Printf ("Modified second linked list v: \n");
Printlist (v);
getchar ();
return 0;
}
```

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

```
#include < stdio.h >
Void find (int ar[100], int n, int s)
{
 int Sum = 0 ;
 int i = 0, h = 0 ;
for (i=0 ; i <n ; i++)
{
while ((Sum < s) && (h <n))
{
   Sum = Sum + ar[h];
   h++ ;
 if (sum==s)
{
  Printf ("found");
   return ;
}
}
Sum = ar [l] ;
}
}
int - main (void)
{
 int ar[100] = {3,4, 7,9,8}
 int s = 25 ;
find (ar, n, 5) ;
return 0 ;
}
```

4) Write a program to print the elements of queue in a :) in reverse order
ii) in alternate order

→ i) Queue in reverse order

// Program to print the element of a queue in reverse order

```
#include <stdio.h>
#include <stdlib.h>
Struct Stackrecord
{
  int *array ;
  int Capacity;
  int tos ;
}
type def struct Stackrecord * stack;
Stack CreateStack (int max)
{
  Stack S ;
  S = malloc (size of (struct Stackrecord ) );
  if (S ==NULL)
{
  Printf (" out of Space" );
}
S → array = malloc (( size of (int ))* max );
if (S → array == NULL)
{
  Printf (" OUtof space" );
}
S→ Capacity = max - 1 ;
S→ tos = -1
```

```
  return(s);
 }
 int isempty(stack s)
 {
  return s->tos==-1;
 }
 int isfull(stack s)
{
  return s->tos==s->capacity;
}
  void push(int x, stack s)
{
   if(isfulls(s))
 Printf("overflow");
 else
 {
  Printf("\n%d is Pushed", x);
   s->test;
   s->array[s->tos]=x;
 }
}
int topandpop(stack s)
{
  if(isempty(s))
{
 Printf("\nempty queue");
 return 0;
}
 else
{Printf("\n%d is popped", s->array[s->tos]);
 return s->array[s->tos--];
}
}
```

```
    if (isfullq(q))
    Printf ("overflows");
    else
    {
     Printf ("\n %d is enqueued", x);
     q -> rear++ ;
     q -> array [q-> rear] = x ;
     if (q -> front == -1)
     q -> front ++ ;
    }
   }
int frontanddelete (queue q)
{
   int P ;
   if (isemptyq(q))
{
   Printf ("underflow");
   return 0 ;
    }
else
{
  P= q -> array [q -> front ];
 Printf ("\n %d is front and delete", P);
 q -> front ++ ;
 return P ;
}
}
Void display (queue q )
{
 int i, rear ;
 if (isemptyq(q))
 {
```

```
struct queuerecord
{
    int *array;
    int front;
    int rear;
    int capacity;
};
typedef struct queuerecord *queue;
queue createqueue(int max)
{
    queue q;
    q = malloc(sizeof(struct queuerecord));
    if(q==NULL)
    printf("Error");
    q->array = malloc(sizeof(int)*max);
    if(q->array==NULL)
    printf("Error");
    q->capacity = max-1;
    q->front = -1;
    q->rear = -1;
    return q;
}
int isfullq(queue q)
{
    return(q->rear==q->capacity);
}
int isemptyq(queue q)
{
    return(q->front==-1);
}
void enqueue(queue q, int x)
{
```

```
printf (" underflow ");
return ;
}
for (i = q -> front ; i <= rear ; i++)
printf (" %d \t", q -> array [i]);
}
int main ()
{
int max, ele, i, choice, n = 0, x, z, capacity;
queue q;
stacks;
printf ("\n Enter the maximum elements");
Scanf (" %d", &max);
q = Createqueue (max);
s = Creati stack (max);
while (1)
{
Printf (" \n menu : 1. Insert 2. Display reversed order 3. exit ");
Printf (" Enter the choice ");
Scanf (" %d", & choice );
Switch ( choice )
{
case 1 :
Printf (" \n Enter the element: ");
Scanf (" %d", & ele );
Enqueue (q, ele);
n++ ;
break ;
Case 2 :
Printf (" \n contents of the queue: ");
display (q);
for (i = 0 ; i < capacity ; i++ )
```

```
Z = front and delete (q), s ;
Push (z, s) ;
}
q → front = -1 ;
q → rear = -1 ;
for (i = 0 ; i < capacity ; i++)
{
Y = top and pop (s) ;
enqueue (q, y) ;
}
Printf ("\n Reversed contents are: ");
display (q) ;
break ;
Case 3 :
exist (0) ;
}
}
}
```

ii) In alternate order

```
#include <stdio.h>
#include <stdlib.h>
Struct node
{
   int data;
  Struct node *next;
};
Void push ( struct node *head_ref, Char new)
{
Struct node *node_new = (Struct node *) malloc (sizeof (struct node));
 node_new -> data = new;
 node_new -> next = (*head_ref);
 (*head_ref) = node_new;
};
  int main ( )
{
  Struct node *Head = NULL;
          Push (&Head, 7);
          Push (&head, 5);
          Push (&head, 3);
          Push (&head, 1);
          Push (&head, 8);
          Print alternate (head);
          return 0;
}
Void print alternate (struct node * head)
{ int count = 0;
  while (head != NULL)
{
```

```
if (count % 2 == 0)
{
    cout << head -> data << " ";
    count ++ ;
    head = head -> next ;
}
}
}
```

---

5) (i) How is array different from linked list

Ans :-

(i) An array is the data structure that contains a collection of similar type data elements whereas the linked list is considered as non-permitive data structure contains a collection of unordered linked elements known as nodes.

(ii) In the array the elements belong to indexes i.e., if you want to get into the fourth element you have to write the variable name with its index or location within the square bracket.

(iii) In a linked list though, you have to start from the head and work your way through until you get to the fourth element.

(iv) Accessing an element in an array is fast, while linked list takes linear time, so it is quite a bit slower.

(v) operations like insertion and deletion in arrays consume a lot of time. on the other hand, the performance of these operations in linked lists is fast.

(vi) Arrays are of fixed size. In contrast, linked lists are dynamic and flexible and can expand and contract its size.

vii) In an array, memory is assigned during compile time while in a linked list it is allocated during execution

viii) Elements are stored consecutively in arrays whereas it is stored randomly in linked lists

ix) The requirement of memory is less due to actual data being stored within the index in the array. As against, there is a need for more memory in linked lists due to storage of additional next and previous referring elements.

x) In addition memory utilization is inefficient in the array. Conversely, memory utilization is efficient in the linked list.

5 (ii) Write a program to add the first element of one list to another list for 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} in list 2

```
# include < stdio.h >
int main ( )
{
  int b1 [100], b2 [100] ;
  int i, v, position, n = 10 ;
  for (i = 0 ; i < 10 ; i++)
  {
    Scanf ("%d", &b1 [i]);
  }
  for (i=0 ; i<n ; i++)
  Printf ("%d ", b [1]);
  Printf ("%d", b [2]);
  v = b2 [0];
  position = 1 ;
  n++ ;
  for (i=n ; i >= position ; i--)
    b1 [i] = b1 [i-1];
    b [position - 1] = v ;
for (i = 0 ; i < n ; i++)
printf ("%d ", b [i]);
for (i = 1 ; i < n ; i++)
Printf ("%d", b2 [i]);
return 0 ;
}
```