A. Chaitanya
AP19110010501

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 the taken from user

Ans:-

```
#include <stdio.h>.
#include <stdlib.h>
  Struct node
  {
  Struct node * next;
  };
  struct node * next;
  Void input (struct node *s)
  void delete (struct node *z)
  Void main(void)
  {
  struct node * s;
  int n;
   s = null;
  do
   {
   Print("Enter the element to in sert : \n");
   print f ("2. Delete\n")
   Print f ("3 exit \n");
   Printf(" Enter the choice? ");
   scanf ("%d", &n);
  switch(n)
  {
   case 1 : input (s);
            break;
   case 2 : delete (s);
            break;
       3 while (n != 3)
   }
   Void input (struct node * z).
   {
```

```
int pos; c=1
 curr = z;
Printf ("Enter the element to be inserted:");
 Scanf ("%d", &pos);
     while (curr -> next ! = Null)
      {
      c++.
      if (c = = pos)
      {
      temp= (struct node*) malloc (size of (struct node));
      Printf = ("Enter the numbers:");
       Scanf = ("%d", &temp -> n);
       temp -> next = curr -> next
       curr -> next = temp.
       break;
       }
     }
  }
 Void delete (struct node* z)
 {
  int pos; c =1
   curr = z!
 Print f ("Enter the element to be delete:");
 scanf ("%d", & pos);
 while (curr -> next ! = Null)
 {
  c++
  if (c = = pos)
  {
  temp= curr -> next,
  curr -> next = (curr->next -> next,
  free (temp)
 }
 curr = curr -> next;
 }
 void merge (struct node* p, struct node *q).
 {
  struct node* p-curr = p, * q - Curr = * q)
 {
```

Scanned with CamScanner

```
Struct node* p-next, *q-next;
while (p-curr != Null && q-curr != Null)
{
    p-next = p-curr->next;
    q-next = q-curr->next;
    q-curr->next = p-next;
    p-curr->next = q-next;
    p-curr = p-next;
    q-curr = q-next;
}
*q = q-curr
}
int main()
{
Struct node *p = Null, *q = Null;
Push(&p, 1);
Push(&p, 2);
Push(&p, 3);
Printf("first linked list :\n");
Print list (p);
Push(&q, 4);
Push(&q, 5);
Push(&q, 6);
Printf("Second linked list : \n");
Print list (q);
Push(&q, 4);
Push(&q, 5);
Push(&q, 6);
Printf("second linked list :\n");
Print list (q);
merge (p, &q);
printf ("modified first linked list :\n");
print list (p);
printf (" Modified Second linked list :\n");
Print list (q);
return 0;
}
```

2) Construct a new linked list by merging alternatives 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}

so

```
#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 * sorted merge (struct node *a , struct node *b)
{
 struct node dummy;
 struct node * tail = & dummy)
 dummy. next = Null;
 while (1)
 {
  if (a = = Null)
  {
   *y = newnode -> next;
   newnode -> next = * x;
    *x = newnode
   }

 }
 void push (struct node ** head-ref, int new-data)
 {
  struct node* newnode = (struct node *) malloc (size of (struct node
  new node -> data = new data;
  newnode -> next = (* head-ref);
  (* headref ) = newnode;
 }
 void print list (struct node *node)
 { while (node ! = Null)
 {
  Print f (" %d ", node -> data);
  node = node -> next;

 }
 },
tail->next=b
break;
}
else if (b=Null)
{
 tail->next =a;
 break;
}
 else if (b=Null)
{
 tail->next=a;
 break;
}
else if (b=Null)
 {
  tail-> next =a;
  break;
}
 tail->next=a;
 break
}
if (a->data <= b->data);
{
more node{ (tail ->next, a);
}
else
 {
  move node ((tail)->next, b));
 }
 tail = tail -> next;
 }
  return (dummy next);
}
void move node (struct node* x, struct node ** y)
{
struct node* newnode = y;
assert (new node != Null);
}
int main( )
{
```

```
struct node *res = null;
Struct node *a = null;
Struct node *b = N=14,
push (&a, 1);
Push (&a, 2);
push (&a, 3);
push (&b, 4);
push (&b, 5);
push (&b, 6);
res = sorted merge (a, b);
printf "("merge linked list is ;\n"),
Print list (res);
return 0;
}
```

3) find all the element in the Clist whose sum is equal to k (where k is given from user)

Sol:
```
# include <stdio.h>
int s1[10], top 1 = -1, s2[10], top2 = -1;
int s1 empty ()
{
 if (top) == -1)
    return 1;
   else
     return 0;
 }
 int s1 pop ()
 {
   top 1 --;
 }
 int s1 [++ top 1] = n;
}  s1 [++ top 1] = n;
}
int s2 empty ()
 {
  if (top == -1)
      return 1;
   else
     return 0,
}
int s2 top ()
{ return s2 (top2);
}
```

```
int s2 pop()
{
  top2--;
 }
  int s2 push(int x)
   {
    s2 [++ top2) = x;
    }
    int sum (int k)
    {
     int n;
    while (s1 empty() != 1)
    {
     int n;
     while (s1 empty() = 1)
     {
       if (n + s1 top () = k)
     {  printf ("% d, %d)\n", n, s1 top();
      }
      s2 push( s1 top()}
        s1 pop();
    }
     while (s2 empty () != 1)
      {
      s1 push (s2 top());
        s2 pop()
       }
     }
   }.
int main ()
{
  int n, i, a, k;
  printf "(enter the no of elements of stack :\n")
   scanf ("%d", &n);
    for (i=0; i<n; i++)
    {
     scanf ("%d", &c);
       s1 push(c);

    }
     push.
     printf("enter the value of constant sum: \n")
scanf ("%d", &k);
~~P~~printf ("The combination whose sum is equal to k is :\n");
sum (0);
} }
```

4) Write a program to print the element in a queue :
 i) in reverse order
 ii) in alternate order

sol

```
1) # include < stdio.h>
# include " stack : h"
# include " QQ . h")
int main ()
 {
  int n , arr(20) , i , j = 0
  struct stack s ;
  int stack (20);
 printf ("Enter no");
 scanf ("%d ", &n);
 for (i = 0 ; i<n , i++)
 { printf ("Enter values : ");
   Scanf ("%d", &arr(i));
 } printf ("Enter value : ");
   Scanf ("%d ,& arr (i);
 } for  (i = 0 , i <n ; i++)
    { insert ( arr( i));
    } while (i) = n)
    } while (i) != n)
    } push (& s, del());
     j++ ;
   } print f ("Reverse is ");
    while ( stra a -1)
```

Scanned with CamScanner

```
}
 Printf ("%.d", pod (u2));
}
 Printf ("\n");
 return 0;

}
Q1)
 #Include <stdio.h>
#  Include <stdlib.h>
 struct node
  {
  int data;
  struct node * next;

 }
 void print node (struct node *head)
  {
   int count =0;
  while (head != Null)
   {
     if (count % 2==0)
     {
       print f ("%d" head -> data);

     }
     Count++;
      head=head->next;

    }
   }
void push (struct node ** head-ref , int new-data).
 {struct node* new node = (struct node*) malloc(size of(structure node))
new-node-> data =new. data;
new node -> next =(*head-ref);
   (*head-ref):=new-node;

 }
 ind main ( )
 {
  struct node * head =null;
  push(*head (12);
Push (& head, 11);
push (& head, 18);
push (& head, 6);
Push (& head, 23);
Print node (head);
return 0;
}
```

5).

i) How array is diff from the linked list?

ii) write a program to add the first 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

Sol:

i) The major difference between array and linked lists regards to their structure, Array, are index data structure where each element associated with an index. on the other hand, linked list relies on reference to the previous and next element.

ii)

```
# include <stdio.h>
# include <stdlib.h>
struct node
{
int data;
   struct node * next;
} void push (struct node** head-ref , int new data)
{
struct node ** new node = (struct node*) malloc
                              (size of (struct node));
   new node-> data = new data
   new node -> next .(& head - ref);
(*head_ref) = new_node
}
void print list (struct node* head)
{
 struct node * temp = head
 while (temp != Null)
  {
   Print f ("%d", temp-> data);
  temp = temp -> next ;
  }
   printf ("\n");

  }.
```