AP19110010415

CSE-H

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

Sol:-
```
#include<stdio.h>
# include <stdlib.h>
struct node
{
   struct node *next;
};
 struct node *curr,*temp;
 void input (struct noder)
 void delete (struct noder)
 void main (void)
{
  struct node *s;
 int n;
  s= Null;
 do
 {
  printf("Enter the element to insert; \n");
  printf ("2. Delete \n");
  printf ("3. Exit \n");
  printf("Enter the choice ; ");
  scanf ("%d", &n);
     switch(n)
   {
```

```
case 1: input (s);
        break;
case 2: delete (s);
        break;
   } while (n!= 3)
}
void input(struct node *z)
{
 int  pos, c = 1
 curr c x;
 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
  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;
  printf ("Enter the element to be delete;"
  scanf ("%d", & pos);
```

```
while (curr → next! = Null)
{
  c++;
  if (c= = pos)
  {
   temp = current → 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;
  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_curr;
   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);
merge (p, &q);
printf ("modified first linked list = \n");
printlist (p);
printf ("modified second linked list = \n");
printlist (q);
return 0;
}
```

2, Construct a new linked list by merging alternatives of two list for example in list 1.We have {1,2,3) & in list 2 we have {4,5,6} in the new list we should have {1,4,2,5,3,6}

Sol:-
```
#include <stdio.h>
# include <stdlib.h>
struct node
{
    int data;
    struct node *next;
};
void move node (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 = new node ->next;
  new node ->next = * x;
   * x = new node;
}
void push ( struct node ** head_ref, int new_data)
{
 ~~whi~~
 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 point list (struct node *node)
{
  while (node $ = Null)
  {
    printf ("%d", node -> data);
    node = node -> next;
  }
}
  tail -> next = b;
   break;
  }
   else if (b = = Null)
   {
     tail -> next = a;
     Break;
   }
  if (a -> data  < = b -> data)
  {
    move 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 *  new node = *y;
  assert (new node ! = Null);
  int main()
  {
   struct node *  res = null;
   struct node *  a = null;
   struct node* b = null;
```

```
    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");
    printf list (res);
    return 0;
}
```

output:- merge linked list is: 1, 4 2 5 3 6

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

```
# include <stdio.h>
int s1[10], top1 = -1, s2[10], top2 = -1;
int s1empty()
{
 if (top1 == -1)
    return 1;
 else
      return 0;
 }
  int s1 top()
 {
  return s1[top1];
 }
  int s1 pop()
 {   top1--;
 }
  int s1 push(int x)
 {
    s1[++top1] = x;
 } int s2 empty()
 {
   if (top2 == -1)
       return 1;
   else
        return 0;
 }
 int s2 top()
 {
   return s2[top2];
 }
  int s2 pop()
```

```
{
  top 2 - - ;
 }
 int s2 push (int x)
 { ~~[illegible]~~
   s2 [+ + top 2] = x ;
 }
   int sum (int k)
   { int x ;
     while (s1 empty () != 1)
   {
       x = s1 top () ;
     s1 pop () ;
   while (s1 empty () ! = 1)
   {
      if (x + s1 top () = k)
   {
      printf ("%d , %d] \n", x, s1 top ());
   }
     s2 push (s1 top ());
      s1 pop () ;
   }
    while (s2 empty () ! = 1)
    { s1 push (s2 top ());
        s2 pop () ;
    }
  }
}
int main()
{
  int n, i, e, k ;
  printf ("enter the no. of elements of stack : \n");
  scanf ("%d", &n);
```

```
for(i=0;i<n;i++)
{
    scanf("%d", &e);
        s1 push(e);
  }
    printf("enter the value of constant sum : \n");
  scanf("%d", &k);
  printf("The combinations whose sum is equal
                              to  k is : \n");

  sum(k);
}
```

output:- Enter the no. of elements of stack :

```
5
6
5
4
9
8
Enter the value of constant sum:
9
The combinations whose sum is equal to k is;
(4, 5)
```

1. WAP to print the elements in a queue
i, in reverse order   ii, in alternate order

sol i,
```
  # include <stdio.h>
  # include "stack.h"
  # include "QQ.h"
int main()
{
  int n, arr[20], i, j=0;
  struct stack s;
  initstack (&s);
  printf ("enter a number");
  scanf ("%d", &n);
  for (i=0, i<n ;i++)
  {
     printf("enter the values:");
    scanf ("%d", & array[i]);
  }
  for (i=0 ;i<n; i++)
  {
    insert(arr[i]);
  }
   while.(j!= n)
  {
   push (&s, del());
   j++;
  }
   print ("Reverse is");
  . while (stop != -1)
  { printf("%d", pop(&s));
  }
   printf ("\n");
retorn0;
}
```

```
ii, #include <stdio.h>
    #include <stdlib.h>
     struct node {
         int data;
          struct Node * next;
       }
       void print nodes (struct Node * head)
       {  int count =0;
         while (head! = Null) {
           if (count % 2 ==0) {
             printf ("%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 (struct node));
      new-node-> data = new-data;
      new_ node -> next = (* head - ref);
      (*head- ref) = new_node;
    }
    int main ( )
    {
        struct mode * head = Null;
          push (&head, 12);
          push (& head, 29);
          push (& head, 11);
          push (& head, 23);
          push (& head, 8);
print node (head);
return 0;
}
```

output:-

```
i, Enter number : 5
   Enter values : 10
   20
   30
   40
   50
ii, Reverse is
      50
      40
      30
      20
      10
iii, head → data
   12   11   10   6   23
   head
   alternative
   12   10   23
```

}

5, i, How array is different from the linked list
ii, WAP 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)
{ printf ("%d", temp->data);
  temp = temp -> next;
} printf ("\n");
}
```

~~out~~put :-

```
data in first linked list: 2 3 4 5
data in second linked list 6 7 8 9
new_data = 2 6 3 7 8 9
                   4     5
```