V Venkat Raju
AP19110010139

①

```
#include <stdio.h>
#include <stdlib.h>
 struct node
{
  struct node * next;
};
Struct node *curr, * temp;
  Void input (struct node*)
   Void delete (struct node*)
   Void main (Void)
 {
  Struct node *s;
   int n
   s= Null
 do
 {
  Printf ("Enter the element to insert: \n");
  Printf (" 2. Delete \n");
  Print f (" 3. Exit \n");
  Print f (" Enter the choice:");
Scan f ("%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 = z;
    Printf ("Enter the element to be inserted: ");
    Scanf ("%d", *pos);
        while (curr -> next != Null)
    {
        c++;
        if (c == pos)
    {
        temp = (struct node*) malloc (sizeof (struct node));
        Printf ("Enter the numbers: ");
        Scanf ("%d", & temp -> n);
        temp -> next = curr -> next
        curr -> next = temp;
        break;
    }
}
```

```
Void delete (struct node * 2)
{
  int POS, c = 1;
   curr = 2;
  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(R);
    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);
```

```
Print f ("modified second linked list =\n");
  Print list (a);
   return 0;
}
```

(2)

```
#  include <stdio.h>
#  include <stdlib.h>
#  include <assest.h>
 struct  node
  {
  int data;
   struct node * next;
  } ;
Voide move node (struct node **x, struct node **y);
   struct node * sorted merge (struct node *a, struct
                                node*b)
{
struct node dummy;
struct node * tail = & dumy;
 dumy . 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)
{
  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)
  {
    Print f ("%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 ** 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");
    Print list(res);
    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>
int S1[10], top1 = -1, S2[10], top2 = -1;
int S1 empty()
{
    if (top1 == -1)
        return 1;
    else
        return 0;
}
int S top()
{
    returns1[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 ( )
{
   top2--;
}
int S2 Push(int x)
{  S2 (++top2] = x
}
int Sum (int x)
```

```
}
  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);
}
```

(4)

(i)

```
# include <stdio.h>
# include <stack.h>
# include "QQ.h"
int main()
{
  int n, arr[20], i, j=0;
  struct stack S;
  init stack (&S);
  Printf("Enter no");
  Scanf("%d", &n);
```

```
for (i= 0, i< n, i++ )
{
 Printf(" Enter values : ");
 scanf ("%.d", & n);
 for (i=0, i<n, i++)
 {
   Printf(" Enter values: ")
   Scanf ("
   insert (arr[i]);
 }
  while (i != n)
 {
  Push (& S, del());
  i++;
 }
 Print f ("Reverse is ");
 while (stop != -1)
 {
  Printf ("%.d ", Pop (& S));
 }
  Print f ("\n");
return 0;
}
```

ii)

```
#include <stdio.h>
#include <stdlib.h>
struct node {
 int nodata
 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 node * head = Null;
  Push (&head, 12);
  Push (& head, 29);
  Push (& head, 11);
  Push (& head, 13);
  Push (& head, 8);
  Print node (head);
return 0;
}
```

(5) (i) The major difference b/w Array and linked lists regards to their structure. Arrays are index based 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
                              ( sizeof ( 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");
}
```