DSA Assignment

S. SRIKAR
AP19110010474
CSE - F

①
```
# include <stdio.h>
# include <conio.h>

struct node
{
  struct node * next;
};
  struct node *s;
  int i;
   s=Null;
  do
  { Printf (" Please enter insert element; \n");
    Printf (" 2. Delete\n");
    Printf (" 3. Exit \n");
    Printf (" Enter your choice.");
    scanf ("%d", &n);
        switch (n)
      {
        case 1: input (s);
                break
        case 2: delete (s);
                break;
      } while (n!= 3)
  }
  void input ( struct node *s)
   {  int  pos, c=1
      ans = 1;
      Printf (" Please enter element");
      scanf ("%d", &s->n);
       while ( ans->next != Null)
       {
```

```
if (c== pos)
{
temp= (struct node*) malloc (size of node(struct));
Printf ("enter the numbers");
scanf ("%d", &temp->n);
temp-> next = cur -> next;
cur ->next = temp;
break;
}
}
void delete (struct node*z)
{
int pos, c=1;
cur =z;
Printf("enter the element to be deleated:\n");
scanf ("%d", &pos);
while (cur -> next!= NULL)
{ c++;
if (c== pos)
{
temp = (cur) -> next;
cur -> next = cur -> next -> next;
free (temp)
}
cur = cur -> next;
}
void swap (struct node *p, struct node*q)
{ struct node *p_cur =p, *q_cur = *q;
struct node *p_next, *q_next;
while (p_cur = NULL && q_cur != NULL)
```

```
{
    q->next = q->curr->next;
    q->curr->next = p->next;
    p->curr->next = q->curr;
    p->curr = p->next;
    q->curr = q->next;
}
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");
    Print list(p);
    Printf("Modified second linked list is:");
    Print list(q);
    return 0;
```

```
{
    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**, struct node**);
{ struct node* newnode = *y;
  assert(newnode!=NULL);
int main()
{
  struct node *res = null;
 struct node *a = NULL;
```

(2)

```
# include <stdio.h>
# include <stdlib.h>
# include <cmath>
struct node
 {
   int data;
 struct node* next;
};
 void move node (struct node **, struct node **);
{
    struct node dummy
    struct node* tail = &dummy
    dummy.next = Null;
    while (1)
     {
      if (a == null)
     { 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 Print list (struct node * node)
```

```
        if (top2 == -1)
            return 1;
        else
            return 0;
}
    int S2 topE()
{   return S2[top2];
}   int S2 PopC()
{   top2 --;
}
    int sum(int k)
{   int x;
    while (S1 empty() != 1)
    {   x = S1 topC();
        S1 PopC();
    while (S1empty() != 1)
    {   if (x+S1 topC() = k)
        {
            printf("%d, %d)\n", x, S1 topC());
        }
        S2 Push (S1topC());
        S1 PoP ();
    } while (S2 empty() != 1)
    {
        S1 Push (S2 topC());
        S2 PopC();
    }
```

```
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)

```
# include(stdio.h)
int S1[10], top1 = -1, S2[10], top2 = -1;
int S1 empty()
{ if (top == -1)
    return 1;
  else
    return 0;
}
int S1 top()
{
  return S1[top];
}
int S1 Push(int x)
{
  S1[++top1] = x;
}
```

```
int main()
{
    int n,i,e,k;
    Printf("Enter the no of elements of structure:\n");
    Scanf("%d",&n);
    for(i=0;i<n;i++)
    {
      Scanf("%d",&e);
       s.push(e);
    }
    Printf("Enter value of constant sum:\n");
    Scanf("%d",&k);
    Printf("The combinations whose sum is equal to
                k is:\n");

       Sum(k);
}
```

//

(ii)

```
#include<stdio.h>
```

(i)

```
#include "stack.h"
#include "q.h"

int main()
{  int ,n, a[20], i, j=0;
   struct stack s;
   initstack(&s);
   Printf("Enter no");
   Scanf("%d",&n);
   for(i=0,i<n,i++)
```

```
}
  Scanf ("%d", &arr[i])
  {  insert (arr[i]);
}
while (i!=n)
{
  Push (&s, del());
  i++;
}
Printf ("Reverse is");
 while (stop! = -1)
{
   Printf ("%d", Pop(&s));
  }
Printf ("\n");
return 0;
}
```

(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) {
```

```
Printf("%d", head->data);
}
  count++;
 head = head->next;
  }
}
void push(struct Node** head_ref, int new_data)
  {
    struct node* new_node = (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, 23);
    Push (&head, 8);
    Print node (head);
   return 0;
}
```

(5)

(i) The differences are :-

* Regarding structure
* Arrays are index based
* Linked list relies on reference to previous of next element.

(ii)

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node *next;
};
void Push (struct node **head_ref);
{
    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 Print list (node *head)
{
```

Print

```
{
  struct node * temp = head;
  while (temp != NULL)
  {
    Print f ("%d", temp->data);
    temp = temp->next;
  }
  Print f("\n");
}
```