Assignment - 4

D. Gireeswar Varma
AP19110010557
CSE-F

1)

```
#include <stdio.h>
#include <stdlib.h>
  Struct node {
    int data ;
    Struct Node *next ;
    } ;
    Struct node * head;
    Void insest (int data, int n) {
     node * temP = new node ;
     temP —> data = data ;
     temP —> next = NULL ;
      if  (n = = -1) {
      temP —> next = head ;
      head  = temP ;
       return ;
     }
    Void delete (int K ) {
     Struct node * temP = head ;
     if   (K = = 1 ) {
     head  =  temP —> next ;
     Free  (temP ) ;
     return ;
     }
     Node * temP = head ;
     For   (int  i=0 ; i<n-2 , i++ ) {
     temP  =  temP —> next ;
     }
```

```
Void Print ( ) ;
For (int i=0 ; i<k-0 ; i++ )
tempP = tempP -> next ;
Free (tempP) ;
}
int main ( ) {
int n, x, k
head = Null ;
PrintF ("enter the Position for inserting") ;
ScanF ("%d", &n ) ;
scanF ("%d", &x ) ;
Insert (x,n ) ;
PrintF ("enter the Position to delete") ;
ScanF ("%d", &k)
delete (k ) ;
PrintF (x ) ;
return ;
}
```

2)

```
# include < stdio.h >
# include <stdio.h >
  struct node {
   int data ;
  struct node * next ;
  }
  Void Print list ("struct node * head )
  }
```

```
Printf ("%d ---> ", (Ptr -> data));
    Ptr = Ptr -> next;
Printf ("Null \n");
}
Void Push (struct node *head, int data)
{
 struct node *new = (struct node *) malloc
  new -> data = data;
  new -> next = *head;
  *head = new;
 }
  struct node *merge (struct node *a, struct node *b)
  {
   struct node Fake;
   struct node *Tail = Fake;
   fake.next = null;
   while (1) {
   if (a == null)
   {
    Tail -> next = b;
    break;
   }
   else if (b == null)
   {
    Tail -> next = a;
    break;
   }
```

```
else
{
 Fail ->next = a
 Fail = a
  a = a ->next ;
  Fail ->next = b;
  }
}
 return Fake.next ;
 }
 int n = size of (keys)
  Struct node *a = null ; *b = null;
  For (int i=n-1; i>0 ; i=i-2)
    Push (&a , key [i]);
  For (int i=n-2; i>=0 ; i=i-2)
     Push (&b ; key [;] );
     Struct node * head = merage (a,b);
     Print F list (head);
   }
```

3)

```
#include <stdio.h>
void Find (int ar[], int a, int k) {
 int total = 0
 int x=0 , y = 0 ;
 For (x=0; x<a, x++ ){
 while (total < k, ++y < a )
```

```
bobal = arr[y]
  y++ ;
if (bobal == 0 )
{
PrintF ("Find");
 return ; }
 bobal -= arr [x] ;
 }
}
  int main (Void ) {
   int arr [] = {9,10, 12,4, 1, 2]
   int k= 565;
   int a - size OF (arr) / size OF (arr(0));
   Find (arr,a,k);
    return 0;
  }
```

4) i)

```
# incude <stdio.h>
 # define max 20
  void show ( int stack [], intsize, int toP )
 {
 int i;
 For (i=0 ; i< size ; i++ )
 {
 PrintF (" in value OR % d is %.d ,toP , stack")
   toP = toP -1 ;
 }
```

```
Void reverse (int stack [], int queue [], int *b, int *r,
{                                                int *f)
   F=0;
  while (b>-1)
  {
     r= *r+1;
    queue [*r] = stack [*b];
     *b = *b+1;
    }
     while (*f <= *r)
     {
       *b= *b+1;
     stack [*b] = queue [*f];
       *f = *f+1;
     {
       topP = topP + 1;
       Printf ("enter value Position is");
       scanf ("%d", &item");
    stack [topP] = item;
    }
     Show (stack, size, topP);
     reverse (stack, queue, &topP, &rear, &Front)
     Printf ("After reverse");
      Show (stack, size, topP);
      getch ();
     }
```

:)

```
# 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;
 }
 {
  Struct node * new-node = (struct node)
      new -node -data =new-data
      new -node -next = (* head -ref);
       (* head -ref) = new -node;
      }
       int main ()
      Push (&head , 1, 2);
      Push (&head , 2 0 );
      Push (&head , 2 3);
      return 0;
    }
```

5) i) How array is different from link list.

Array

1) An array is collection of element of similar data type.

2) Array element can be accesed randomly in array index

3) Data element are stored in cantinuous location in memory.

linked list

1) linked list is collection of element of same type in each element correct using pointers.

2) Random accessing is not possible in link list element with accesed seqvenly.

3) New element can be sorted any where & reference is created for new element using pointers.

ii)

```
# include <stdio.h>
# include <stdlib.h>
 Struct node
{
 int data;
 struct node *next;
```

```
}
void Push { struct node ** head - ref data }
{
  Struct node * new node = { Struct node * ) malloc
   new -node -> data = new -data )
   new -node -> nex C : head -ref );
      ( * head - ref ) = new node
   }
   Void Print list (Struct node * head )
   {
     Struct node = stemP = head ;
     while C temP = null )
     {
     Print F (" % d", temP -> data )
       temP = temP -> next .
     }
     Print F (" \n" / ; )
     }
```