Assignment-4

K. pavan teja
Ap19110010315

1)
```
# include   < studio.h >

# include <studio.h>
    struct Node {
     int data{
      struct Node " next;
    };
      struct Node head;
   void Insert (int data, intn){
   Node * temp= New Node ( ) ;
    temp -> data = data;
    temp -> next = Null;
    if (n==1) {
      temp -> next = head;
            head = temp :
            return;
        }
      Void   Delete -(int) () {
     struct  Node * temp = head;
      if  (K==1){
       head = temp -> next;
         return;
     }
       Node  * temp = head;
  for (int i =0;  i< n-2, i ++){
```

Scanned with CamScanner

```
kemp  =  .kemp -> next ;

}
   kemp  -> next  =  temp+next;
   temp  -> next  =  temp';

}
     void  print( );
  for (int  i=0, i< K-2, i++)
      temp =  temp -> next;
         free (temp)',

       }
      int main ( ) {
        int n, x, K;
         head= null',
    printf ("enter the position for and inseoting;")
         scanf ("%d "&n);
         scanf ("%d"&x);
         Insert(x, n);
    printf (" enter  the position to delite);
         scanf  (" %d "& K);
         Delete( );
         point (x);
            return;
          }
```

②
```
# include <studio.h>
#  Include < studio.h>
    stuct node{
        int data;
     stuck node* next;
    }
     void print list (stuct node* head)

    {
       printf("%d ->" (ptr->data));

          ptr = ptr-> next;}
        print f("null/n");
         }
        void push (stuct node* head ,int data)

         {
           stuct node * new= (stuct node*) malloc
             (size of (struct node));
              new-> data= data;
               new-> next = *head;
                 * head= new;

             }
               struct node * merge(stuct node * a.stuct
             node * b
          } struct node fake;
         struct node* fall= fake;
          fake .next = Null;
          while (1){
         if (a == Null)
          { fail -> next= b;
              break;
        }
```

```
else if(b = null)
{
    tail -> next = a,
  break;
}
    else;
{   ~~tail ->~~
    tail -> next = a,
        tail = a
         a = a -> next!
        tail -> next - b!,
}
}
    return   take next!,
}
    void main()
{
    int Keys[ ] = { 1,2,3,4,5,6,7}
  int n = size of (Keys) / size of Key[0]
   struct node  *  a = null * b = null,
for (int i = n-2,  i>=0,  i = i-2)
  push(s b: Key[i]);
struct node  *  head = merge (a,b)!
 print list (head);
}
```

(3)

```
#include <studio.h>
void find (int arr[ ], int a, int k) {
        int total = 0
        Int x = 0,  y = 0!
    for  (x = 0, x < a, x ++ ){
while (sum<17, -y<a)
        arr [y)!
     y + + :
            for (x=0 ;x<; x++){
    while ( total<k , && y<a)

            total   = arr [y]
                y ++;
            if (total = =0)

            {  printf(" find");
              return;}
              total   = arr [x],

          }
          }    int main (void){

int arr[ ] = {9,10,12,4,1,2,3}

          int k =  565;
        int a = size of (arr) /size of arr (0);

              find (arr,a,k);
                returno;

            }
```

④ i) Reverse order   ii) Alternate order

```
        # include <stdio.h>
        # define size 20
        void   insert  (int);
         void  delete ();
    int queue [20], a = -1, b = -1;
          void main (){
   int  num, choice;
   while (1){
 printf ("\n"menu \n");
  print ("1.insert \n2 Delete\n3 print
            n4. Reverse\n4 alternate\n5 exit);
   print f( \n" enter your choice");

   Scan f  ( "%d", & choice ");
```

Scanned with CamScanner

```
                insert (num);
                break;
printf ("Reverse queue");
        for(int i =size, i>0; i--)
          if (queue [i]= = 0]
        continue;
          printf ("%d", queue [i]);
      }
          break;
case 3 :-   printf ("Alternative elements");
           for( int i =0 ,i<size, i>0, i++2)
        {  if |queue [i] = =0)
             continue;
              printf ("%d", queue(x));
           } ~~bea~~ break;
          return0;
          }
```

(5) i) Arrays vs linked lists

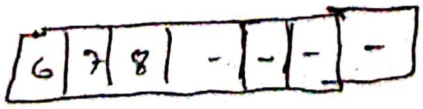
1) Both are the data structures, both are used to store the data

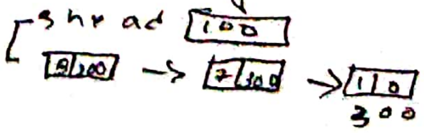
2) lost of accessing the elements.

memory Requirment & utilization

Array	linked list
Inffective in memory utilization.	→ it it is dynamic size

Ex.





8 v 3 = 24 bytes

→ more = requirments.

it takes of constant time 8 x u = 8 = bytes

⇒ Require meory in less

iv) cost of insertion and cast of deletion

Array		linked list
Begining - (o)n	-	o(1)
At end - (o)1	-	o(n)
ith position - o(n)	-	o(n)

v) easy use and operations

Array	linked lists
eaiser to use	=> less eaiser
linear and binary	linear

vi)
```
# include < studio.h>
#  include <studio.h>
  int ~~[illegible]~~ len[int a()]
{ int   i=0 , x,y = 0;
      while (1)
   {
      if   (a[i])
   {
      xy ++, i++;
   } else
   { break;
   }
      return xy;
} void change list (int x[ ], int a[ ])
{  for (int i= len[x]-1, i>=0, i--)
{   x[i+1] = x(i];
}
   x[o] = a[a];
printf (" /n elements of add array : /n")
    for (int i=0; i<len(x) ; i++)
    { printf (" %d", x(i));
  }
  for (int i=0, i<(only); i++)
  {  y [i]= y(i+1);}
  printf ("/n element of new array ; /n")
    for( int i=0; i<len (a); i++)
    {  print f ("%d", a[i]);
  } int main()
 {  int x [10]= {1,2,3} , a[10] ={4,5,6});
        Change list =(a,b);
    }
```