) Write a program to incert and delete an element at the nth and kth pocition in a linked Lict where n and k is taken from user.

A)
```
#Include <stdio.h>
#include <stdlib.h>.
struct node {
 int data;
 struct Node*next;
 };
 struct node*head;
 void incest(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;
 }
temp->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 inserting:");
  scanf("%d",& n);
  scanf("%d",& x);
 Insert(x,n);
  printf("Enter the position to delete);
  scanf("%d", & k)
  delete(k);
```

```
print (x);
return ;
}
```

2) Construct a new linked list by merging alternate nodes of two lists for example in list 1 {1,2,3} and in list 2 {4,5,6} in new list we should have {1,4,2,5,3,6}

A)
```
# 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
              (size of (structnode ));
  new -> data = data ;
  new -> next = * head;
   *head=new;
```

```
}
struct node* merge(struct node*a, struct noble *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
   tail = a
   a = a -> next ;
   tail -> next = b ;
   }
 }
 return fake. next ;
 }
 void main()
 {
```

```
int keys[] = {1, 2, 3, 4, 5, 6, 7}
int n= size of (keys)/size of keys[0].
Struct node * a = null ; *b = null;
for (int i=n-1; i>0; i=i-a).
   push(&a, keys[i]);
for (int i=n-2; i>=0; i=i-2)
   push(&b; keys[i]);
struct node * head = merge (a;b);
 print list(head);
}
```

3) Find all the elements in the stack whose sum is equal to k.

A)
```
# include <stdio.h>
 void find(int arr[], int a, int k){
  int total =0
  int x=0, y=0;
  ~~for (x=0; x<a, x++)){~~
  ~~while~~ for (x=0; x<a, 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};
   int k = 565 ;
   int a = sizeof(arr)/sizeof(arr[0]) ;
   find (arr, a, k);
   return 0;
 }
```

4) A)

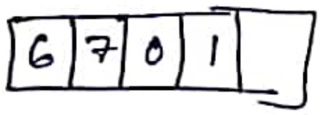
```
# include <stdio.h>
 # define size 20
 void insert(int);
 void delete () ;
 int queue [20], a = -1, b = -1;
 void main() {
 int num ; choice ;
 while(1) {
 printf("\n New \n");
 printf("1.insert \n 2.delete \n 3. Print
        \n 4.Reverse \n 5.Alternate \n 6 Exit");
   printf("\n Enter your choice");
scanf("%d", &choice);
switch (choice) {
case 1: printf("Enter the num to insert:");
  scanf("%d", &num);
  insert(num);
 break;
case 2: printf("Reverse queue");
     for (int i = size, i≤0, i--)
       if (queue[i]==0)
         continue;
     printf("%d", queue[i]);
   }
     break;
case 3:
     printf("Alternate elements:");
       for (int i=0, i<size, i>0, i++2)
   { if (queue[i]=0)
         continue;
     printf("%d", queue [i]);
   }
     break;
   return 0;
  }
```

5) i) Array vs linked lists

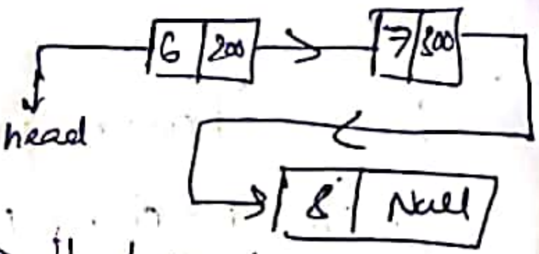
1) Both are the data structure .. Both are used to store the data.

2) Cost of accessing the elements

Arrays



linked list



⇒ it takes at constant time

$\boxed{O(1)}$

⇒ It depends on number of nodes in the linked list

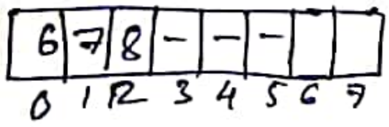
$\boxed{O(n)}$

3) Memory requirement and utilization.

Array

⇒ Ineffective in memory utilization

Ex:

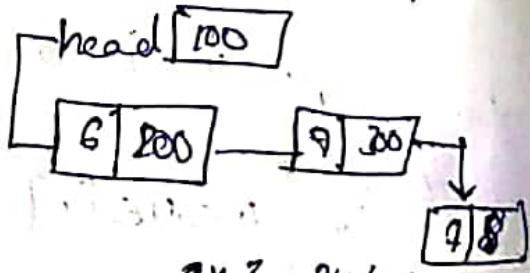


8×4 = 32 bytes

Used = 12

linked list

⇒ it is in dynamic byte



8×3 = 24 byte

⇒ Require memory in less

⇒ More requirements

4) Cost of insertion and cost of deletion

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

5. Easy use and operations

Array	linked list
→ easier to use	→ less easier
→ linear and binary	→ linear

(ii)
```
#include <stdio.h>
 #include <stdlib.h>
 int len[int a()]
 {
  int i=0, x,y=0
   while(1)
   {
    if (x[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[0] = a[0];
  printf("\n Elements of old array :\n");
 for (int i=0; i<len(x); i++)
  { printf("%d", x[i]);
  }
 for (int i=0, i<len(y); i++)
   {
      y[i] = y[i+1];
   }
 printf("\n Elements of newarray: \n")
 for (int i=0; i<len(a); i++)
{printf("% d", a[i]);
}
int main()
```

```
{int x[10]={1,2,3}, a[10]= {4,5,6};
  Change list=(a,b);
}
```