Assignment - 4

P. Manoj
AP110010054
CSE-G

1) Write a programme to insert and delete an element and nth and kth pointer in linked list where n and k are taken from user.

A)
```
#include <stdio.h>
#include <stdlib.h>
struct Node {
int data ;
struct mode * head ;
void Insert (int data ,int n) {
Node * temp = new mode ;
temp -> data = data;
temp -> ~~data~~ 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;
    Print f("Enter the Posistion for and
                                   insurting:");
     scanf("%d,"&n);
    scanf("%d": & x);
     Insert (x,n).
```

```
Point f ("Enter the position to delete);
scanf ("%d", & k);
Delete (k);
Print (x');
return;
}
```

2) Construct a new linked list by merging alternative nodes and two lists for example inthe list 1 we have {1,2} & list 2 {4,2,0} and in the new we should have {1,4,2,0,3,6

A)
```
# include <stdio.h>
# include <stdlib..h>
struct node {
        int data;
      struct node *next)
  }
  void Print list (struct node * head)
 {
  Print f (·"%d ->", (Ptr-data));
   Ptr = Ptr -> next:}
   Print f ("Null/n");
 }
  void Push (struct node *head ,int ddi)
 {
```

```
struct node *new= (struct node*) malloc
        (size of (struct node));
new -> data =data;
new -> next= *head;
 * head = new;
}
struct node * merg (struct node *a, struct
                                  node *b)
{
  struct node fake;
 struct node * fail= fake;
 fake = next= Null;
 while (1){
if (a== Null)
 {
  tail ->next=b;
 break;
}
 else if (b= Null)
{
 fail -> next=a;
  break;
} else.
}
 fail -> next =a;
  fail=a;
 a= a-> next;
```

Scanned with CamScanner

```
    fail -> next = b;
  }
}
  return false. 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-1, i >0; i = i-a)
    Push (& b; . Key[i]);
 struct node * head = merge (a,b);
  Print list (head);
}
```

③ Find all elements in start whose sum is equal to k

Ⓐ
```
  # 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 ( sum < k , & y <a)
         = arr[y]
       y++;
```

Scanned with CamScanner

```
for (x=0; x<a; x++){
  while ( total < k; a * 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= sizeof (arr)/sizeof arr(0));
 find (arr ,a,k);
 return 0;
}
```

4) write a Programme to Point elements to Queue?

i) Reverse Order ii) Alternate Order

```
# include <stdio.h>
#
   define size 20
```

```
void insert (int);
void delete ();
int arr [20], a= -1, b=-1;
void main(){
int num ; choice;
while (1){
Print f ("\n "new" \n");
Print f("1. insert \n 2. Delete \n 3 Print
n4 Reverse .\n 4. Alternative \n 5 : exit);
Print f ( \n Enter your choice ");
scan f (" %d", & choice ");
switch h ( choice ){
case1: Print f ("Enter the num to insert ");
scan f (" %d ", & num);
insert (num);
break;
case 2:
     Print f (" Reverse queue");
  for (int i= size , i>0; i --)
  if (queue [i]=0]
     continue;
  Print f ("%d ", queue(i]);
}
break;
return 0;
}
```

1) i) How array is different from linked list,

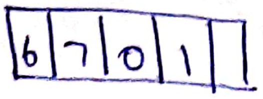
2.) write a programme to add first element of one list to another list for example we have (1, 2, 3) in list 1 and (4, 5, 6) in list 2 we have to get (4, 2, 3) list 1 and (5, 6) list 2.

i) Arrays vs linked lists

1. Both are data structures. Both are used to store the data.

2. Cost of accessing the elements.

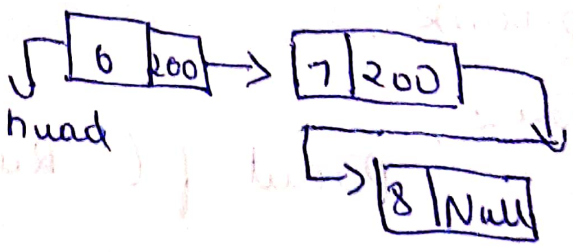
Arrays



⇒ it takes at constant time

$\boxed{O(1)}$

linked lists



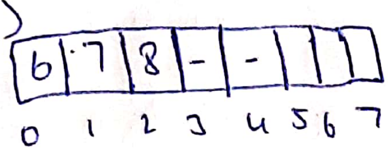
⇒ it depends on number of nodes in the linked list

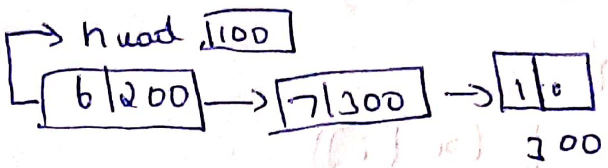
$\boxed{O(n)}$

3. Memory Requirment and utilization

Array	linked list
=> Inflation in memory utilization	=> it is in dynamic size

Eg)





8×4= 32 bytes	8×3=24 bytes
used =12	
=> Require memory	=> more requirement in less

4) Cost of insertion and cost of deletion

Array	linked list
Begining= O(n)	O(1)
At end - O(1)	- O(n)
i th position - O(n)	- O(n)

5. Easy use and operations

Array	linked lists
=> ~~at~~ easier use	=> less easier
=> linear and Binary	=> linear.

(iii)

```
# include <stdio.h>
# include <stdlib.h>
int len[int a[]]
{ int i=0, x,y=0;
while (1]
{
 if (x[i])
{
 x y++, 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];
Print f (u/n elements of old . array :/n
for (int i=0; i<len (x); i++)
{ Print f (" %d ", x[i]);
```

Scanned with CamScanner

```
}
for (int i=0 , i< len (y) ; i++)
 {   y[i] = y [i+1]; ]
     Print f ("\n elements 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 [0] = { 4,5,6} );
     change list = (a,b);
 }.
```