Assignment

Shaik.Meenaz
Ap19110010556
CSE-H

1. Write a program to insert and delete an Element at the nth and kth pointer in a liked list where n and k are taken from the user.

A
```
#include <stdio.h>
# include <stdlib.h>
Struct node {
int data ;
Struct Node* Next ;
};
Struct Node* ~~Next~~ head;
void Insert (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, 7, k;
 head = Null;
 printf (" Enter the position for and inserting: ");
```

Scanned with CamScanner

```
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 alternative nodes and two lists for Example in list L we have {1, 2, 3} and list {4, 5, 6} and in the new we should have {1, 4, 2, 5, 3, 6}

A
```
# include< stdio.h>
 # include < stdlib.h>
 struct node{
     int data;
   * struct node next,
 }
 void printlist(struct node*head)
 {
 printf("%d→"(ptr→ data));
 ptr= ptr→next;}
```

```
printf ("Null/n");
}
void push(struct node* head,int data)
{
 struct node knew=(structnode*)malloc
            (size of (structnode));
  new→data = data;
  new → next=* head;
   * head= new;
  }
  Structnode* 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
{
  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)
 struck 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 (&h; key (j]);
 struct node * head = merge (a, s);
```

Scanned with CamScanner

```
printlist (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 (sum<k, && y <a)
              = arr(y);
          y++;
 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[] = {9,10,134,1,2,3}
   int k= 565;
  int a= size of (arr)/ size of (arr[0]);
  find(arr, a, k);
   return 0;
}
```

4.) Write a program to print elements of Queue?

i, Reverse order

ii, Alternate order

```
# include <stdio.h>
# define size 20
void insert (int);
```

Scanned with CamScanner

```
void delete ();
int queue [20], a=-1, b=-1;
void main(){
int num; choice;
while (1) {
printf("1.Insert\n2.Delete\n3.print\n4.Reversal
        \n4.Alternate\n5.. Exit");
printf("Enter your choice");
scanf("%d",&choice");
Switch (choice){
Case1:printf("Enter the numtoinsert");
 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, p<size; i>0, i++2)
 {
  if (queue [i]==0)
  continue;
  print f("%d", queue [i]);
 }
  break;
 return 0;
  }
```

5) i) How arrayis different from liked list?

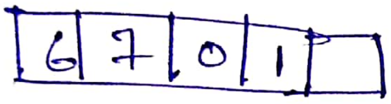
2.) Write a program to add first element of one list to another list for Example we have (1, 2, 3) in list 1 and (4, 5, 6) in list 2 we have to get (4, 1, 2, 3) as output for list1 and (5, 6) list 2.

ii, <u>Arrays vs Linked lists</u>

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

2. Cost of accessing the Elements.

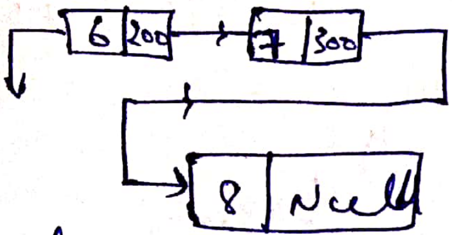
<u>Arrays</u>



⇒ it takes at constant time

$\boxed{O(1)}$

<u>Linked Lists</u>



⇒ It depends on number of nodes in the linked lists

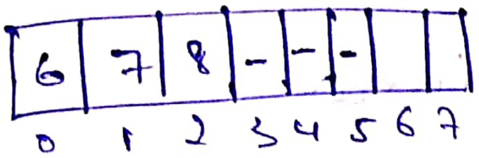
$\boxed{O(n)}$.

3 <u>Memory Requirement and utilization</u>

Array

⇒ Inffective in memory utilization

Ex

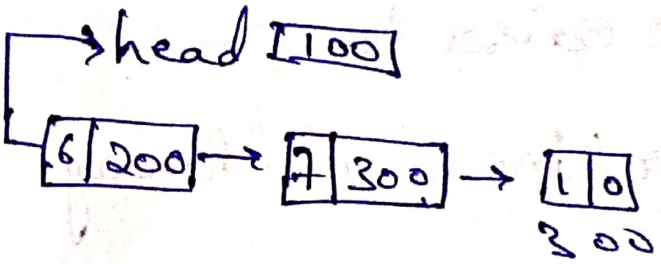


8×4=32 bytes
used=12

⇒ Requires memory in less

Linked lists

⇒ It is in dynamic size



8×3=24 bytes

⇒ More requirement.

4. cost of intersection 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
⇒ eaiser to use	⇒ less eaiser
⇒ Linear and Binary	⇒ linear

ii, # include<stdio.h>

```
# include <stdlib.h>
int len[int a()]
{
int i=0, x, y=0;
while (1)
{
if (1[i])
{
xy++, i++;
}
else
{
```

```
break;
}
}
return xy;
}
void changeList (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++)
 {
```

Scanned with CamScanner

```
y[i]= y[i+1]; }
printf("\n Elements of new array:\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 Cid = (a, b);
}
```