Assignment - 4

Vaishnavi.D
AP19110010374
CSE-H

1. Write a program to insert and delete an element at the n^{th} & k^{th} position in a linked list where n and k are taken from the user.

```
#include <stdio.h>
#include <stdlib.h>
struct node
int data;
struct node * next;
};
struct node * head;
void insert (int data, int n;
Node * temp = new node n;
temp → data = data;
temp → next = null;
if (n==1){
temp → next = head;
head = temp;
return;
}
void delete - (int n){
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,a,k;
head = null;
pf ("Enter the position for and e inserting:");
scanf ("%d"&n);
scanf ("%d",&a);
Insert (a,n);
printf ("Enter the position to delete);
Scanf ("%d",&k);
delete (*);
print (a);
return;
}
```

2. Construct a new linked list by merging alternative nodes of 2 lists. for ex: in list 1 we have {1,2} & list 2 {4,2,6} and in the new we should have {1,2,3,4,5,6}

```
#include <stdio.h>
#include <stdlib.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) mall (size of struct node);

new→data = data;
new→next = *head;
*head =new;
}
struct node *mege (struct node* a, struct node* b)
{
struct node pate;
stud node *fast = fake;
fake = next = null;
```

```
while(1){
if (a == Null)
{ tail → next = b
break;
}
else if (b=null)
{
tail → next = a;
break;
}else
{ tal → next = a
 tail = a
  a = a → next
 tail → nex - 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-2 ; i >= 0; i = i - a)
  push (&a) key[i]);
for (int i = n-2, i >=0, i = i-2)
  push(& b; key [j]);
struct node * head = merge(a,b);
 print list (head);
}
```

3. find all the elements in the stack 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++;
for(x=0, x<a; x++){
while (~~sum<k && y<a~~) (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);
   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);
Void delete();
int queue[20], a=-1, b=-1;
Void main(){
int queue[20],a=-1,b=1
Void main(){
int num; choice;
while(1){
printf("\n" new"\n");
printf("1 insert\n2. Delete\n3. print n4. Reverse\n4
    allenate\n5. Exist);
printf("\n Enter your choice");
scanf("%d",&choice");
switch(choice){
Case1 : printf("Enter the numb 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] == -d)
        continue;
         printf("%d", queue[i]);
       }
        break;
       return 0;
       }
```

5. (i) How array is different from linked list?

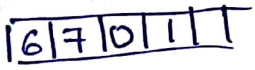
2. Write a program to add first element of one list to another list for example we have (1,2,3) in list 1 & (4,5,6) in list 2 we have to get (1,2,3) as output for list 1 and (5,6) list 2

(i) Arrays v.s linked lists

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

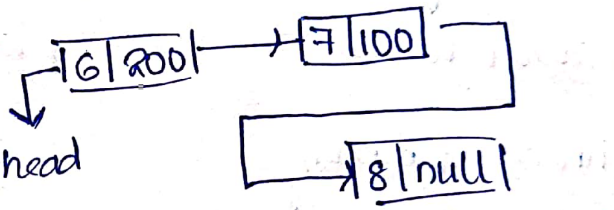
2. Cost of accessing the elements

Arrays



⇒ It takes at constant time

Linked list



→ It depends on no. of nodes in the linked list O(n)

3. Memory Requirement & utilization

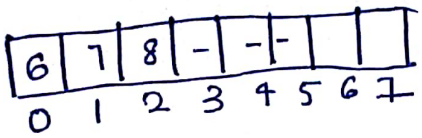
Array

⇒ Ineffective in memory utilization

Linked list

⇒ It is in dynamic size

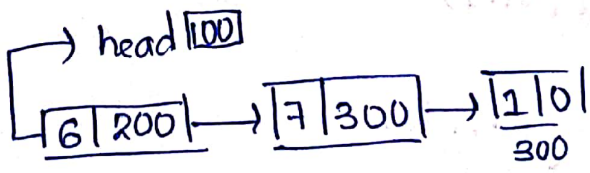
Eg:



8 x 4 = 32 bytes

Used = 12

⇒ Require memory is less



8 x 3 = 24 bytes

⇒ More requirement

4. Cost of insertion and cost of deletion

Array		Linked list
Beginning - O(n)	–	O(1)
At end - O(1)	–	O(n)
i^{th} position - O(n)	–	O(n)

5. Easy use and operations

Array	Linkedlist
⇒ easier to use	⇒ less easier
⇒ linear & binary	⇒ linear

(ii)
```
#include <stdio.h>
#include <stdlib.h>
int len[int a[]]
{
 int i=0, a1y=0;
 while(1)
 {if[a[i])
 {
  ay++; i++;
 }
 else
 {
  break;
 }
 }
 return ay;
}
void change list (int a[5], 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 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 list = (a, b);
 }.
```