Assignment - 4

P. Siddharth Roy
AP19110010527

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

A)

```
# include <stdio.h>
# include <stdio.h>
Struct Node {
int data;
Struct Node * next;
};
Struct Node *head;
void Insert (int data, int n){
Node *temp = newNode (n);
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 i
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 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 alternative nodes and two lists for example in list1 ~~an~~ we have {1, 2} and list 2 {4, 2, 6} and in the new we should have {1, 4, 2, 5, 3, 6}

A)

```
#include <stdio.h>
# include <stdio.h>
Structnode {
    int data;
    Struct node *next;
}
void print list (structnode* head)
{
Printf ("%d →", (Ptr → data));
    Ptr = ptr → next;
    Printf ("Null/n");
}
void push (struct-node *head, int data)
{
Structnode *new.(struct node *de malloc
                (size of structnode);
new → data = data;
new →next = head;
* head = new;
}
Structnode *merge (struct-node *a, structhode *b)
```

```
{
 Struct node fake;
 Struct node * fail = fake;
    fake.next = Null;
    while(1) {
 if (a == Null)
 {
   fail → next = b;
   break;
  }
  else if (b == null)
  {
   fail → next = a;
   break;
 }
 else
 {
   fail → next = a;
    fail = a;
    a = a → next
   fail → next = b;
  }
}
return fake.next;
}
void main()
{
 int keys[] = {1,2,3,4,5,6,7}
Int n = size of (keys) / size of key[0]
```

```
Structnode *a = Null ; *b = Null;
for (int i = n-1, 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 whoose 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)
{
```

```
{
  Print f ("find ");
   return; }
   total -= arr[i];
  }
}
int main (void) {
int arr (9) = {9,10,12,4,1,2,3}
  int k = 565;
int a = size of arr)/size of arr(0)];
  find '(arr, 9, k)
  return 0;
}
```

---

4) Write a program to print elements to Queue?

i) Reverse order ii) Alternate order

```
# include <stdlonh>
# define size 20
void insert (int t);
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 ("In" "new" In");
```

```
Printf ("1. Insert \n2. Delete \n3. Print \n4. Reverse \n4 Alternate
                    \n5. Exit");
Printf ("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;
}
```

---

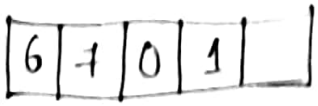
5) 1) How ~~many~~ array is different from linked list

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 (4,1,2,3) as output for list 1 ; and (5,6) in List 2.

i) Arrays Vs linked lists

1. Both are the data structures. Both are used to store the data.
2. Cost of accessing the elements

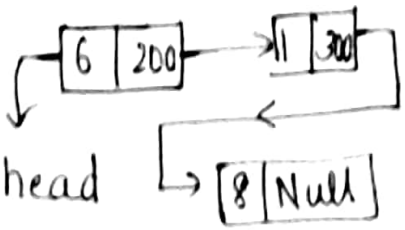
Arrays



=> it takes at constant time

O(1)

linked list



=> It depends on number of nodes in the linked list

O(n)

3. Memory Requirment and Utilization

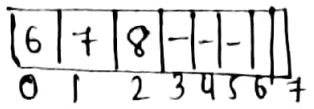
Array

=> Infective in memory utilization

linked list

=> It is in dynamic size.

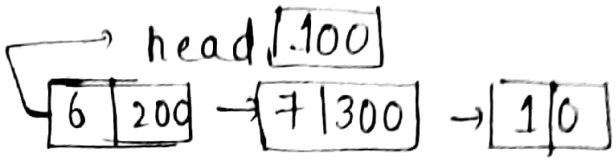
Ex:-



8 x 4 = 32 bytes

Used = 12

=> Require memory in less



8 x 3 = 24 bytes

=> More requirement

4. Cost of insertion and Cost of deletion.

	Array	linkedlist
Begning	O(n)	O(1)
At end	O(1)	O(n)
ith position	O(n)	O(n)

5. Easy use and operations

Array	linkedlist
=> eaiser to use	=> less easier
=> linear and binar	=> linear

ii) #include <stdio.h>

include <stdio.h>

int len(int a[])

{

```
    int i=0,x,y=0;
    while(1)
    {
     if(x[i])
    {
      xy++, i++;
    }
    else
    {
      break;
    }
}
   return xy;
}
 void changeslist(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 ("In elements of new array: In")
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);
}
```