Page No. ______
Date:

Assignment :- 4

Q.1) write a program to insert and delete an element at the nth and kth position in a linked list where n and k is taken from the user

Ans)
```
# include<stdio.h>
# include<stdlib.h>
Void ans (node *, int, int)
int size = 0;
struct node {
int data ;
struct node * next;
}
node *get node (int data)
{
node * newnode = (struct node*) malloc(newnode)
newnode -> data = data ;
newnode -> next = null;
return newnode ;
}
void ans (node * Current, int pos, int data ).
printf (pos<1 || pos >size +1 )
printf ("Invalid");
else
{
while (pos--)
{
if (pos == 0 ).
{
node * temp = getnode (data );
temp -> next = * Current ;
* Current = temp ;
}
```

```
else
{
current = &(*current)->next;
}
size++;
}
}
void printf(struct node* head);
{
    while (head != null)
    {
    printf("%d", head->data);
        head = head->next;
    }
    printf("\n");
}
void del (struct node* head def, int pos) {
 if (head-ref = null)
    return;
    temp = head-ref;
    if (pos=0)
    {
    *head-ref = temp->next;
    free (temp);
    For (int i=0; temp != null && T<pos-1; i++)
    temp = temp->next;
    free (temp->next);
    temp->next = next;
    }
    int main ()
    {
    struct node* head = null;
    push (& head, 7);
```

```
push (&head, 8);
push (&head, 6);
ins (&head, 7, 15);
del (&head, 4);
print list (head);
return (0);
}
```

Q2) Construct a new linked list by merging alternate nodes of two lists for example in list 1 we have {1,2,3} and in list 2 we have {4,5,6} in the new we should have {1,2,3,4,5,6}

Ans
```
# include <stdio.h>
# include <stdlib.h>
struct node {
    int data;
    struct node* next;
};
void print list (struct node * head)
    while (ptr)
    {
    printf ("%d → ", ptr → data);
    ptr = ptr → next; }
    print f ("null \n");
}
void push (struct node * head, int data)
{
struct node * new = (struct node *) malloc (size of
                                    (struct node));
new → data = data;
new → next = * head;
```

```
*   head = new ;
}
struct node * merge (struct node * a, struct node
                                        * b)
{
  struct node dummy;
  struct node * tail = dummy;
  dummy.next = Null;
    while (1) {
   if (a == Null)
   {
     tail -> next = b;
      break;
   }
    elseif (b = Null)
    {
      tail -> next = a;
       break;
    }
    else
    {
      tail -> next = a;
       tail = a;
      a = a -> next;
         tail -> next = b;
  }
  }
  return dummy.next;
  }
   void main ()
  {
  int keys [] = {1,2,3,4,5,6,7};
  int n = size (keys) / size of key [0];
```

```
struct node *a = null, *b = Null;
for (int i = n-1; i>0; i = i-2).
    push (&a, keys[i]);
for (int i = n-2; i>=0; I = i-2).
    push (&b, keys (I));
structe node* head = merge (a,b);
    print list (head);
}
```

Q3) Find all the elements in the stack whose sum is equal to k (where k is given by the user.

Ans
```
# include <stdio.h>
void find (int arr[], int n, ints) {
  int sum = 0;
  int l = 0, h = 0;
  For (l=0; l<n; l++) {
    while (sum<s && h<n)
      sum &t = arr (h];
        h++;
    if (sum == s)
    {
      print ("found");
       return; }
    sum = arr (l];
  }
}
int main (void) {
  int arr[] = {2,6,0,9,7,3}
    int & = 15;
  int n = size of (arr) / size of (arr[0]);
    find (arr, n, s);
      return 0;
}
```

Qu) write a program to print the elements in a queue:
(i) in reverse order (ii) in alternate order

Ans
```
# include <stdio.h>
# include <stdlib.h>
struct node
{
  int data;
  struct node * next;
}
void print rev(struct node * head)
{
  if (head = NULL)
    return;
  print rev(head -> next);
  print f("%d", head -> data);
void push (struct node * head rev, cha new)
{
  struct node *node-new = (struct node* )
                 malloc (sizeof (struct node))
  node-new -> data = new;
  node - new -> next = (head* - ref);
  (* head.ref ) = node-new;
}
int main ()
  struct node * head = NULL;
  push (& head, 4);
  push (& head, 3);
  push (& head, 2);
  print new (head); print alternate (head);
  return 0;
}
```

```
void print alternate (struct node * head)
{
  int Count == 0;
  while (head != NULL)
  {
    if (Count % 2 == 0)
      Count << head -> data << " ";
        Count ++;
    head = head -> next;
  }
```

5] How array is different form linked list.

Answers
key difference between Array and linked list.

1] An array is data structure that contains a collection of similar type data elements whereas the linked list is considered as non-primitive data structure Contains a collection of unordered linked elements known as nodes.

2] In the (elements) array the element belong to indexes i.e.., if you want to get into the fourth element you have to write the variables names with its index & location within the square bracket.

3] In a linked list through, you have to start form the head and coorks your way through until you get to the fourth element.

4] Accessing an element in an array is fast, while in linked list takes linear time, so it is squite a bit slower.

5] operation like insertion and delation in array Consume a lot of time. on the other hand the performance of these operations is linked list is fast.

6] In a array, a memory is assigned during Compile time while in linked list it is allocated during execution of runtimes.

5)(ii)

```
# include <stdio.h>
# include <stdlib.h>
int len (int a[])
{
int i=0, a n=0;
while (1)
{
if (a[i])
{
an++, i++;
}
else
{
break;
}
return an;
}
void changinglist (int a[], int b[])
{ for(int i = len(a) - 1; i >=0 ; i--)
{
a [i+1] = a(1);
}
a(0) = b[0];
printf ("/n the elements of first array: /n ");
for (int i=0; i< len(a) ; i++)
{
printf ("%d", a[i]);
}
for (int i=0 ; <len(b) ; i++)
{
b(i) = b(i+1); }
printf ("/n the elements of second array: /n");
```

```
for (int i=0 ; i<len(b) ;i++)
{  printf("%d ", b[i]);
   }
}
int main ()
{
int a[10] = {1, 2, 3}, b[10] = {4, 5, 6} ;
   changing (int (a, b);
}
```