Assignment - 4

S. Navya
AP19110010414
CSC - H

1. 1. Write a program to insert and delete an element at the nth and k-th position in a linked list where n and k is taken from the user.

```
# include <stdio.h>
# include <stlib.h>
 void ans (node *, int, int)
  int size = 0;
  Struct node {
  int data;
  Struct dat node * next;
  }
  node * get node (int data)
  {
      node * newnode = (struct node * ) malloc (newnode)
       new node -> data = data;
       new node -> next = null;
        Return new node;
   }
    void ins (node * current, int pos, int data)
   {   if (post < 1 || posl > size + 1)
        printf ("Invalid");
      else
      {
```

```
While (pos--)
{
    if (pos==0)
    {
     node * temp = get node (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_uef == null)
return;
```

```
temp = head - uef ;
if (pos = 0)
{
* head - uef = temp -> next ;
free (temp) ;
return ;
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 ) ;
printlist ( head ) ;
 return (0) ;
}
```

2. 2. 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, 4, 2, 5, 3, 6}.

```
# include <stdio.h>
# include <stlib.h>
Struct node {
    int data ;
    Struct node * next ;
}
void printlist ( Struct node* head )
{  Struct node * ptr= head ;
   While (ptr )
   {  printf (" %d —>", ptr->data ) ;
      ptr = ptr —>next ; }
      printf (" 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 * fail = dummy ;
   dummy . next = Null ;
```

```
        push(&a, keys[i]);
    for(int i= n-2; i>=0; i= i-2)
        push(&b, keys[j]);
    Struct node * head = merge(a,b);
    print list(head);
    }.
```

3. Find all the elements in the stack whose sum is equal to K (Where K is given by the user).

```
# include <stdio.h>
void find ( int arr[ ], int n, int s) {
    int sum = 0;
    int l = 0, h=0;
    for(l=0; l<n; l++) {
      While ( sum<s && h<n)
      sum += arr[h];
       h++;
    if (sum == s)
    {   printf(" found");
        &return; }
        sum == arr[l];
  }
}
   int main (void) {
```

```
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 dummy_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-2)
```

```
int arr[ ] = { 2, 6, 0, 9, 7, 3 }
int a = 15;
int n = sizeof (arr) / sizeof (arr[0]);
find (arr, n, S);
return 0;
}
```

4. Write a program to print the elements in a queue.
(i) in reverse order (ii) in alternate order.

```
# include <stdio.h>
# include <stlib.h>
Struct node
{ int data;
  Struct node * next;
}
void printrev (Struct node * head)
{ if (head == Null)
    return;
  print rev (head -> next);
  printf (" %d ", head -> data);
void push (Struct node * head rev, char new)
{ Struct node * node - new = (Struct node *) malloc
                              (Size of (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);
    &eturn 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(i) How array is different from the linked list

Key differences between Array and linked list

1) An array is a 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 array of elements belong to indexes, i.e., if you want to get into the fourth element you have to write the variable name with its index or location within the square bracket.

3) In a linked list through, you have to start from the head and work your way though until you get to the fourth element.

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

5) operations like insertion and deletion in array consume a lot of time. on the other hand the performance of these operations in linked list is fast.

6) In a array, memory is assigned during compile time while in linked list it is allocated during

execution of run time.

(ii)

```
# include <stdio.h>
# include <stlib.h>
Struct node
{
   int data;
   Struct data * next;
 {
  void push (struct node ** head-ref,
                              int new-data)
 {
    Struct node * new_node = (struct node*) malloc
                          (size of (Struct node));
  new_node -> data = new-data;
  new-node ->next = (head-ref);
  (* head-ref) = new_node;
 void print list ( struct node * head )
 {
    struct node * temp = head;
    while (temp! = Null)
    {
       printf ("%d", temp -> data);
       temp = temp ->next;
    }
    printf ("\n");
}
```