DSA - Assignment - 4. Chandu Ch.
#AP19110010111.

Q. Write a program to insert and delete an element at the nth & kth position in a linked list where n & k 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);
new node → data = data;
new node → next = null;
return new node;
}
void int (node* current, int pos, int data)
{
if (pos < 1 || pos > 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 print(struct node * head)
{
while (head != null)
{
printf("%d", data);
head = head->next;
}
printf("\n");
}
void del(struct node * head, int pos)
{
if (head->ref == null)
return;
temp = head->ref;
if (pos == 0)
{
* head->ref = temp->next;
free(temp);
return;
for (int i = 0; temp != NULL && i < 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. 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}

Ans:-

```
#include <stdio.h>
#include <stdlib.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 * tail = dummy;
  dummy. next = NULL;
   while (1) {
   if (a == NULL)
  {
   tail -> ext = 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 keys[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]);
  struct node * head = merge (a,b);
  printlist (head);
}
```

3. 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; 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 n; }
  sum - = arr[l];
}
```

```
int main(void){
  int arr[] = {2,6,0,9,4,3}
  int x = 15;
  int n = sizeof(arr)/sizeof(arr[0]);
  find(arr, n, x);
  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<stdlib.h>
  Struct node
  {
    int data;
     struct node * next;
   }
   void printrev(struct node * head)
  {
   if (head = NULL)
    return;
     printrev(head->next);
     printf("%d", head->data);

void push(struct node* head rev, char 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. (i) How array is diff from the linked list.

Ans: Key differences between Array & linked list

(i) An array is a data structure that contains a collection of similar type data elements where as the linked list is considered as non-primitive data structure contains a collection of unordered linked

elements known as nodes.

2) In the array the 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 & work your way through until you get to the fourth element.

4) According to 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 & 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 times.

(ii)

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

```
for (int i = 0; i < 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 list(a,b);
}
```