(1) Write a program to insert & delete an element at the nth position in a linked list where n & k are taken from 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( );
  temp -> data = data;
  temp -> next = NULL;
  if(n==1) {
  temp -> next = head;
  head = temp;
  return;
  }
  Node *temp = head;
  for (i=0; i<n-0; i++)
  {
    temp= temp->next;
  }
  temp -> next = temp -> next;
  temp -> next = temp;
}   }
```

```
void delete (int r)
{
  Struct Node * temp = head;
  if (r==1)
  {
    head = temp -> next;
    free(temp);
    return;
  }
  Node * temp = head;
  for (i=0; i<n-2; i++) {
    temp = temp -> next;
  }
  temp -> next = temp -> next;
  temp -> next = temp;
}

int main( )
{
  int n,x,k;
  head = NULL;
  printf("Enter position and element to insertion: ");
  scanf("%d", &n);
  scanf("%d", &x);
  insert(x,n);
  printf("Position of deletion: ");
  scanf("%d", &k);
  delete(k);
  print(a)
  return;
}
```

(2)

```
#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)
  {
   struck node *new = struct node *malloc (size of(struct node));
    New ->data = data;
    new ->next = *head;
     *head =new;
    }
Struct node *merge (struck node *a , struct node *b)
{
   struck Node temp;
   Struct Node *tail = &temp;
    temp.next = NULL;
   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 (head->next);
}
 void main( )
 {
  int keys[ ] = {1, 2, 3, 4, 5, 6, 7}
   int n = sizeof(keys) / sizeof key[a]
   Struct node *a = NULL; *b = NULL)
   fo (i = n-i, 1 > 0; i = i-a)
   push (&a, key ([i]);
   fo (i = n-2; i >= 0 ( = i - 2),
    push( &b, key[i]);
   Struct node *head = mige (a, b);
   printlist(head);
 }
```

(8)

```
#include <stdio.h>
int top = -1;
int a;
char stack[5];
void push(int a);
char pop();
int main()
{
int i,n,a,t,k,f,sum=0,count=0;
printf("Enter the number of elements in stack");
scanf("%d",&n);
for(i=0;i<n;i++){
printf("Enter next element")
scanf("%d",&a);
push(a);
}
printf("Enter the sum to be checked");
scanf("%d",&k);
for(i=0;i<n;i++){
[redacted]
t=pop();
sum=t;
count++;
if(sum==k){
for(int j=0;j<count;j++)
printf("%d",stack[j]);
f=1;
break;
}
push(t);
}
if(f!=1)
printf("Elements in stack dont add up to sum");
}
void push(int a)
{
if(top==99)
{
```

```
    printf("\n Stack is full!\n");
    return;
  }
   top = top+1;
    Stack[top]=x;
 }
  char pop()
 {
  if(Stack[top]==-1)
  {
     printf("\n Empty Stack\n");
      return 0;
   }
  x = Stack[top];
 top = top-1;
 return x;
}
```

(4)

```
#include <stdio.h>
# define SIZE 10
void insert(int);
void delete();
int queue[10], f = -1, r = -1
void main()
{
int value, choice;
while()
{
printf("\nmenu\n");
printf(" 1. Intention \n 2. Deletion \n 3. Reverse \n 4. Alternate
Printf (" \n Enter choice:");
Scanf (" %d", &choice);
Switch(choice)
{
case 1: printf("Enter value to insert: "),
        Scanf ("%d", &value);
        insert (value);
        break;
case 2: delete();
        break;
case 3: printf(" Reverse queue:");
        for (i = SIZE; i >= 0; i--)
        {
          if(queue[i] == 0
          continue
          Print ("%d", queue[i]);
        }
        break;
Case 4: printf ("Alternate elements of queue:");
        for (i=0; i < size; i+=2)
        {
          if (queue[i] == 0)
          continue;
          Printf (
```

```
    case 5 : exit (0);
    default : printf ("\n wrong selection! Try again");
    }
}}
void insert (int value){
 if ((f == 0 && r == SIZE-1) || f == r+1)
 printf ("\n Queue is full");
 else {
  if (f == -1)
   f = 0;
   r = (r+1) % SIZE;
   queue[r] = value;
   Printf("\n Insertion success!");
}}
void delete()
{
 if (f == -1)
 printf ("\n Queue is empty");
 else
 {
  printf ("\n Delete : %d", queue[f]);
   f = (f+1) % SIZE;
   if (f > r) f = r = -1;
 }}
```

```
    case 5 : exit (0);
    default : printf (" \n Wrong Selection! Try again! ");
    }
}
void insert (int value){
 if ((f == 0 && r == SIZE - 1) || f == r+1)
 printf (" \n Queue is Full");
 else {
  if (f == -1)
   f = 0;
   r = (r+1) % SIZE;
   queue[r] = value;
   printf(" \n Insertion Success! ");
}
void delete()
{
 if (f == -1)
 printf (" \n Queue is Empty");
 else
 {
  printf (" \n Delete : %d", queue[f]);
   f = (f+1) % SIZE;
   if (r == -1)
 }}
```

(5)

i- How array is different from the linked list?

the major difference b/w Array and Linked list regard to their structure. Array is an index based data structure where each element associated with an index. On the other hand, linked list relies on reference to the previous and next element.

ii.

```
#include<stdio.h>
#include<stdlib.h>
Struct Node
{
 int data;
 Struct node * next;
};
void push (struct Node ** head_ref, int new_data)
{
 struct node * new_node =(struct node *) malloc (size of
                                                  Struct
 new_node -> data = new_data;
 new_node -> next = (*head_ref);
 * head_ref = new_node;
}
void printlist (struct node * head)
{
  struct node *temp = head;
  while (temp != NULL)
  { printf (" %d", temp -> data);
    temp = temp -> next;
  }
  print (" \n");
}
```