```
# 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 ins (node* Current, int pos, int data
{
  if (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");
  }
```

```
Print list(head);
return(0);
}
```

2)

```
#inculde <stdio.h>
#include <stdlib.h>
struct node {
  int data;
  struct node* next;
}
void print list (struct
{
  Structnode *ptr = head;
  while(ptr)
  {
    print f("%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;
}
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)
      Push(&a, keys[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)

```
# include < stdio.h>
int top = -1;
int x;
char stack[100];
void push(int x);
char pop();
int main()
{
int i, n, a, t, k, f, sum = 0, count = 1;
Printf(" Enter the number of elements in the 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++)
{
t = pop();
Sum += t;
Count++ = 1;
if(Sum == k){
for(int j=0; j<count; j++)
Printf("%d", stack[j]);
f = 1;
break;
}
Push(t);
}
if(f != 1)
Printf(" the elements in the stack do not add
up to the Sum");
}
```

```
void push(int x)
{
if (top==99)
{
printf ("\n stack is FULL!!! \n");
return;
}
top=top+1;
stack[top]=x;
}
char pop()
{
if(stack[top]==-1)
{
printf("\n stack is EMPTY!!! \n");
return 0;
}
x=sfrack[top];
top=top-1
    return x;
}
```

```
#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);
    printf("%d", head → data);
void push(struct node* head rev, char new
{
  struct node* node_new = (struct node*) mall
                          (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); printalternate(head);
   return 0;

}
void print alternate(Struct noab *head)
{ int count = 0;
  while (head != NULL)
  {
    if (count%2 == 0)
       count << head → data << " ";
       count++;
       head = head → next;

  }
```

Key differences between Array and linked list:

* 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 struure contains a collection of unordered linked elements known as nodes.
* 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.
* In a linked list through, you have to start from the head and worle your way through until you get to the fourth element.
* Acessing an element in an array is fast, while in linked list takes linear time, so it is quite a bit slower.
* operations like inseation and deletion in array consume a lot of time. On the other hand the performance of these operations in linked list is fast
* In a array, memory is assinged during compile time while in linked list it is allocated during excutim of run time.

```
#include <stdio.h>
#include <stdlib.h>
int len(int a[])
{
  int i=0, an=0;
  while(1)
  {
    if(a[i])
    {
      an++; 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 first array: \n");
for(int i=0; i<len(a); i++)
  {
    print f("%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++)
  {
    print+("%od", b[i]);
    }
}
}
int main()
{
  int a[10] = {1,2,3} ,b[10] = {4, 5, 6};
  Changing list(a,b);
}
```

```
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 & 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);
```