1.
```
#include <stdio.h>
#include <stdlib.h>
void ins (node *, int, int)
int size = 0;
struct node {
int data;
struct node *next;
};
node * new node = new node();
node * nget node (int data)
{
  node *new node = (struct node *) malloc (new node);
  new node -> data = data;
  new node -> next = null;
  return new node;
}
void ins( node * cutout , 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 print(struct node* head)
{
 while(head!=NULL)
 {
  printf("%d", head->data);
  head = head->next;
 }
 printf("\n");
}
void del(struct node *head_ref, 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<position-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);
    delete(&head, 4);
    printList(head);
    return(0);
}
```

2.
```
#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;
  }
  print ("NULL\n");
}
void push(struct node* head, int data)
{
 strude Node *new = (struct node*) malloc(sizeof (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 = dumy;
 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
  tail=b;
   b=b->next;
  }
 }
 return dummy->next;
}
void main()
{
 int keys[]={1,2,3,4,5,6,7};
 int n=sizeof(keys)/sizeof(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, key[i]);
```

```
struct node *head = merge(a,b);
printlist(head);
}
```

3) #include <stdio.h>

```
void find(int ar[], int n, int s)
{
  int sum = 0;
  int l=0 , h =0;
  for( l=0 ; l<n; l++)
    {
      while (sum< s && h<n)
        {
          sum += ar[h];
          h++ ;
```

```
if (sum == s)
{
printf("found");
return;
}
sum -= ar[b];
}
}
int main (void)
{
int ar[] = {2, 6, 0, 9, 7, 3};
int s = 15;
int n = sizeof(ar) / sizeof(ar[0]);
find(ar, n, s);
return 0;
}
```

4.
```
#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_ref, 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);
   printrev(head); printalternate(head);
   return 0; }
```

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

5) i) Array is a set of similar data objects stored in sequential memory allocation under variable name but linked list is a data structure which contains a sequence of the elements where each element is linked to its next element.

ii)
```
#include<stdio.h>
int main()
{
int a1[100], a2[100];
int i, x, pos, n=10;
for(i=0; i<10; i++)
    scanf("%d",&a1[i]); scanf("%d",&a2[i]);
for(i=0; i<n; i++)
    printf("%d",a[1])
    printf("%d",a[2])

x=a2[0];
pos=1;
n++;
for(i=n; i>=pos; i--)
    a1[i]=a1[i-1];
```

```
a[pos-1]=x;
for(i=0;i<n;i++)
    printf("%d",a[i]);
    printf("%d",a
for(i=1; i<n;i++)
    printf("%d",a[i]);
return 0;
}
```