9) write a function "insert_any()" for inserting a node at any given position of the linked list. Assume position starts at

```
insert_any (int item, int key)
{
 struct node *new, *ptr;
 ptr = start;
 if (ptr -> data! = key)
 {
 ptr = ptr -> link;
 }
 else
 }
 new = (Struct node*) malloc
 (Size of (Struct node));
 new -> link = ptr -> link;
 new -> data = item;
 ptr -> data link = new;
 return;
 }
 }
```

9) write a function "delete beg()" for deleting a node from the beginning of the linked list.

```
delete_beg()
{
Struct node* ptr;
if (start = = NULL)
 {
 printf ("IN LIST EMPTY!!");
 return;
```

```
    }
    else
    printf("\n Deleted element is :%d",start
    ptr = start -> link;

    start -> link = NULL;
     start = ptr;
     return;
     }
```

q) write a function "delete - end()" for deleting a node from the end of the linked list

```
delete - end()
 }
 struct node * ptr, * ptr1;
 ptr = start;
```

```
if (start == NULL)
{
printf("\n List Empty!!");
return ;
}
else
}
while (ptr -> link != NULL)
}
ptr1 = ptr;
 ptr = ptr -> link;
 }
 ptr1 -> link = NULL;
 }
 ptr -> link = NULL;
 }
   printf("\n Deleted element is
          :%d", * ptr);
 }
```