**Chapter 7**

**R 7.1 -Give an algorithm for finding the second-to-last node in a singly linked list in which the last node is indicated by a next reference of None.**

**if \_\_name\_\_ == '\_\_main\_\_':**

**l = SinglyList()**

**for i in range(22): #we can just use this 22 size list as example (user can give us anything bigger than or equal to size of 2) I just did not want to do a function**

**l.add\_tail(i)**

**l.\_current = l.\_head**

**z = l.\_head**

**while z != None:**

**x = l.\_current.\_element**

**z = l.\_current.\_next.\_next**

**l.\_current = l.\_current.\_next**

**print(x)#second to last node**

**R 7.3- Describe a recursive algorithm that counts the number of nodes in a singly linked list**

**def nodes(link,pointer): #we assume that we are given a SinglyList**

**if pointer == None:**

**return 0**

**link.\_current = pointer**

**return 1 + nodes(link,link.\_current.\_next)**

**if \_\_name\_\_ == '\_\_main\_\_':**

**l = SinglyList()**

**for i in range(22):**

**l.add\_tail(i)**

**x = l.\_head**

**print(nodes(l,x))**

**R 7.6- Suppose that x and y are references to nodes of circularly linked lists, although not necessarily the same list. Describe a fast algorithm for telling if x and y belong to the same list.**

Given the memory value of x, keep iterating to the next memory address until either you come across y or x itself; whichever comes first. If y comes first, x and y belong to the same list. If you do not come across y then x and y do not belong to the same list.

(I did not code this because I would have to construct a circular list class, but the question only requires an algorithm)

**C 7.28- Describe a fast recursive algorithm for reversing a singly linked list.**

**def reverse(l,pnode,cnode):**

**l.\_current = cnode**

**if l.\_current == None:**

**l.\_head = pnode**

**l.\_current = l.\_head**

**return True**

**else:**

**x = l.\_current.\_next**

**if l.\_current == l.\_head:**

**l.\_current.\_next = None**

**else:**

**l.\_current.\_next = pnode**

**return reverse(l,cnode,x)**

**l = SinglyList()**

**for i in range(10):**

**l.add\_tail(i)**

**reverse(l,l.\_head,l.\_head)**

**for e in l:**

**print(e)**