p.85 3.6

Write a function to add two polynomials. Do not destroy the input. Use a linked list implementation. If the polynomials have M and N terms, respectively, what is the time complexity of your program?

```
typedef  struct  Node  *PtrToNode;
struct  Node  {
      int  Coefficient;
      int  Exponent;
      PtrToNode Next;
};

typedef  PtrToNode  Polynomial;
  /* Nodes are sorted in decreasing order of exponents.*/

Polynomial  Add( Polynomial  a,  Polynomial  b )
{      /* return a polynomial which is the sum of a and b*/
       /*use linked list with a head node */
     PtrToNode  front,  c;
     int sum;
     c = malloc(sizeof(Node));
     if ( c==Null)
          FatalError( "The memory is full");
      front = c;   front->Exponent=-1;
     a=a->Next;   b=b->Next;
     while ( a && b )
          switch ( COMPARE(a-> Exponent, b-> Exponent) ) {
              case -1:     Attach(b-> Coefficient, b-> Exponent, &c);
                           b = b-> Next;
                           break;
              case 0:      sum = a-> Coefficient + b-> Coefficient;
                           if ( sum ) Attach(sum, a-> Exponent, &c);
                           a = a-> Next; b = b-> Next;
                           break;
              case 1:      Attach(a-> Coefficient, a-> Exponent, &c);
                           a = a-> Next;
                           break;
          }
         /* copy rest of list a and then list b */
     for ( ; a; a = a-> Next )   Attach(a-> Coefficient, a-> Exponent, &c);
     for ( ; b; b = b-> Next )   Attach(b-> Coefficient, b-> Exponent, &c);
     c->Next = NULL;
     return front;
}
```

```
void   Attach( int   coefficient,   int   exponent,   PtrToNode   *ptr )
{         /* create a new node, attach it to the node pointer to by ptr.*/
          /* ptr is updated to pointer to this new node */
          PtrToNode temp;
          temp = malloc(sizeof(Node));   /* Do not destroy the input !!!*/
          if ( temp==Null )
                FatalError("The memory is full");
          temp-> Coefficient = coefficient;
          temp-> Exponent = exponent;
          (*ptr)->Next = temp;
          *ptr = temp;
}
```

Time complexity O(m + n)

p.86 3.12

a. Write a nonrecursive procedure to reverse a singly linked list in O(N) time.
*b. Write a procedure to reverse a singly linked list in O(N) time using constant extra space.

Answer to part b:

```
typedef   struct   Node   *PtrToNode;
typedef   PtrToNode   List;
typedef   PtrToNode   Position;
struct   Node
{
    ElementType Element;
    Position Next;
};

List   Reverse( List   L )
/* return a reverse linked list of L, with a header*/
{
     Position   P, Q, R;
     R=NULL;   P=L->Next;
     while (P)
     {
          Q=P->Next;
          P->Next=R;   R=P;   P=Q;
     };
     L->Next=R;
     return L;
}
```

Typical mistake:

```
ptr    a;
a = malloc(sizeof(Node));
a->next = L->next;
L->next = L->next->next;
a->next = NULL;
```