Polynomial.h

複製運算子與複製建構子的部分是將來源多項式從頭開始，將值寫入目標多項式，取代原本的數值，若目標多項式較來源多項式長，則將多餘節點存入 av_list，若目標較短，則在後新增節點。

加法與乘法運算直接將第三次作業的函式改寫為適用鏈結串列的函式，乘法的部分若該冪次項重複則直接改寫數值。

ChainNode.h

經測試發現無法直接改寫 av_list.h 內的 new 與 delete 運算子，且發現若在類別中宣告，則會優先使用類別內宣告的運算子，因此參考第八次作業的範例，改寫成自 av_list 中取出節點與將結點存回的運算子

Av_init 則是將 av_list 依照輸入檔內容初始化為 N 個節點，使用 malloc 函式的原因是若使用 new 運算子，會使用剛剛宣告的 new 運算子，無法達成從記憶體空間宣告節點的目標。

Chain.h

解構子的部分依照 av_list 是否有節點來區分。若有，則將整條鏈結串列接到 av_list 後面，若無，則將 av_list 的 first 與 last 兩個指標以目前鏈結串列的對應指標取代，直接將整條 av_list 用目前串列取代。

Pop 函式則是傳回第一個節點的位址並將之自串列刪除，主要是 av_list 的操作使用。

兩個 InsertBack 函式基本相同，都是將新的元素或結點接到串列之後，若是空串列，則新增第一個節點，將 first 與 last 指標都指向第一個節點。

InsertInorder 函式則是依照由大到小的順序，從頭走訪並將元素插入相對應的位置，若為空串列，操作同 insertBack