# Homework 2

How to do?

用連結串列來儲存能有效解決一開始空間分配的問題

加法的部分邏輯為比大小比較大的就記錄下然後往後走一樣就+一起然後憶起往後

乘法的方式為把比較多的那一方作為基準一項一項*

## 效能分析

### 加法運算 **Add()**:

O(n_1 + n_2)

### 乘法運算 **Mult()**:

O(n_1^2 * n_2^2)

空間消耗分析

1. 結構體本身的大小:
    - `float exp`: 4 bytes
    - `int coef`: 4 bytes
    - `Term* top`: 8 bytes（在 64 位系統上指標佔用 8 bytes, 32 位系統為 4 bytes）
2. 總計:**16 bytes/**節點(在 64 位系統上)。
3. 記憶體額外開銷: 每個 `Term` 是動態分配的, 因此會有額外的分配開銷, 通常是:
    - 內部配置開銷(由 `malloc` 或 `new` 管理):視作業系統和分配器而定, 約 8~24 bytes。

4. 總開銷估算：每個節點的總空間開銷約為：
    - 161616 bytes (資料) + 配置開銷 (8 248~248 24 bytes)
    - 單節點估算：**24~40 bytes**
5. 多項式的總空間複雜度：若有 nnn 個節點：
    - 單一多項式記憶體使用：O(n)O(n)O(n)
    - 若涉及多個多項式（如兩個輸入與一個結果），總記憶體使用量為 O(n1+n2+n3)O(n_1 + n_2 + n_3)O(n1+n2+n3)。

# 測試與驗證

## +法


## *法


# 心得

這次的功課讓我重新複習了連結串列跟動態記憶體配置。