前言：

第一周的課程著重在時間複雜度 (big O) 及空間複雜度 (Ω) 的觀念建立，並提及分治法 (Divide & Conquer) 與遞迴並用的一些經典算法，作業是利用 **Karatsuba** 實現大數乘法。

特殊乘法 —— Karatsuba 演算法：

Karatsuba 算法是世界上第一個比標準算法還快的方法，利用分治法，將兩個數字 x 及 y 的大數乘法，拆成多個位數一半的乘法，再以遞迴的方式，漸進拆解成個位或兩位數的乘法。

現考慮兩個數字 x = 5678 及 y=5678 以標準算法相乘：

```
          5  6  7  8
    X     1  2  3  4
    -----------------------------
       2  2  7  1  2
    1  7  0  3  4

 1  1  3  5  6

 5  6  7  8
-------------------------------
 7  0  0  6  6  5  2
```

由上式可以發現，標準乘法的時間複雜度約為 $O(n^2)$ 。

Karatsuba 發現，若我們將 x 及 y 各拆成兩個等位數的數字 a, b 與 c, d：

x = 5678 → a = 56, b = 78

y = 1234 → c = 12, d = 34

兩數相乘則可依下列步驟計算：

1. a x c = 672
2. b x d = 2652
3. (a+b) x (c+d) = 134 x 46 = 6164
4. 3.-2.-1. = 2840
5. $672\,x\,10^4+2652+2840\,X\,10^{(4/2)}=7006652$

也就是說，兩數相乘可以利用以上 1 ~ 4 步組合後求得，詳細數學公式如下：

$$x \cdot y = 10^{n} x\, ac + 10^{(n/2)} x(ad + bc) + bd$$

上式以程式來實現，可分為三個遞迴解：

1. 遞迴計算 ac
2. 遞迴計算 bd
3. 遞迴計算 (ad+bc) = (a+b) x (c+d) - ac — bd。

時間複雜度：

依 master method，假設兩個數皆為 n 位數且 n 為 2 的冪次，每次遞迴都將位數拆成一半：

$$T(n) \leq 3 \cdot T(n/2) + cn + d$$

因此 Karatsuba 的時間複雜度為 $O(n^{(\log_2^3)}) = O(n^{1.58})$ 。

作業——以 Karatsuba 算法實現大數乘法：

問題描述：

請問以下兩個 64 位數字的相乘結果為何 ？

x = 3141592653589793238462643383279502884197169399375105820974944592

y = 2718281828459045235360287471352662497757247093699959574966967627

解題方法：

題目很好心的將數字設計成 2 的冪次，對於多數使用分治及遞迴的演算法，當輸入資料長度為 2 的冪次時，會有最佳的計算效率，Karatsuba 也不例外。

實際算法設計上，需考慮以下問題：

1. 兩數長度不同，且非 2 的冪次的可能性，在前處理及計算過程中，補零是關鍵步驟。
2. python 的整數沒有實際邊界，所以可利用字串處理補零及正負號問題後，轉成 int 計算。若以 C 或 C++ 等程式語言編寫，由於位數過多，無法以內建的整數資料型態，勢必得以字串的方式表達，需額外處理字串加法、減法及乘法的演算。