**作業系統第一次程式作業程式說明檔案**

**使用開發環境：**  
使用軟體: Dev C++、Virsual Studio(2019) //win10

使用語言:C++，完成了第一題、第二題與第四題的功能，三題未無完成。

程式一開始會先讀取檔案，若檔案不存在則輸出”檔案無法開啟或不存在!”，辨別輸入檔案的的第一個數字，用於但判斷該進行何種任務，若不存在該種任務便會輸出”不存在該種任務，請重新設定檔案”。

第一題，讀入檔案確定任務後直接用讀入的資料進行Bubble Sort排序，並將執行時間印出。

第二題，讀入檔案確定任務後，詢問使者需要用多少Thread，將使用者出入的數字存成K，並把讀入的資料切成K份且每份個別使用Thread進行Bubble Sort排序，完成後用K-1個Thread 進行Merge Sort合併為與讀入黨相同大小的資料，計算程式執行時間，並顯示結果於螢幕，然後將整理完的資料寫入輸出檔，檔名為”輸入檔檔名\_進行的任務編號\_output”。

第四題，讀入檔案確定任務後，詢問使者需要將檔案切成幾份，將使用者出入的數字存成K，並把讀入的資料切成K份後用一個Process使各筆資料個別進行Bubble Sort排序，完成後進行k-1次Merge Sort合併為與讀入黨相同大小的資料，計算程式執行時間，並顯示結果於螢幕，然後將整理完的資料寫入輸出檔，檔名為”輸入檔檔名\_進行的任務編號\_output”

**結果分析：**

將一、二、四題分別執行一萬筆，十萬筆，五十萬筆，一百萬筆資料的檔案的結果秒數，單位為mS，第二和第四題固定切成20筆資料，所有任務的執行時間都隨著檔案資料的增加而增加。

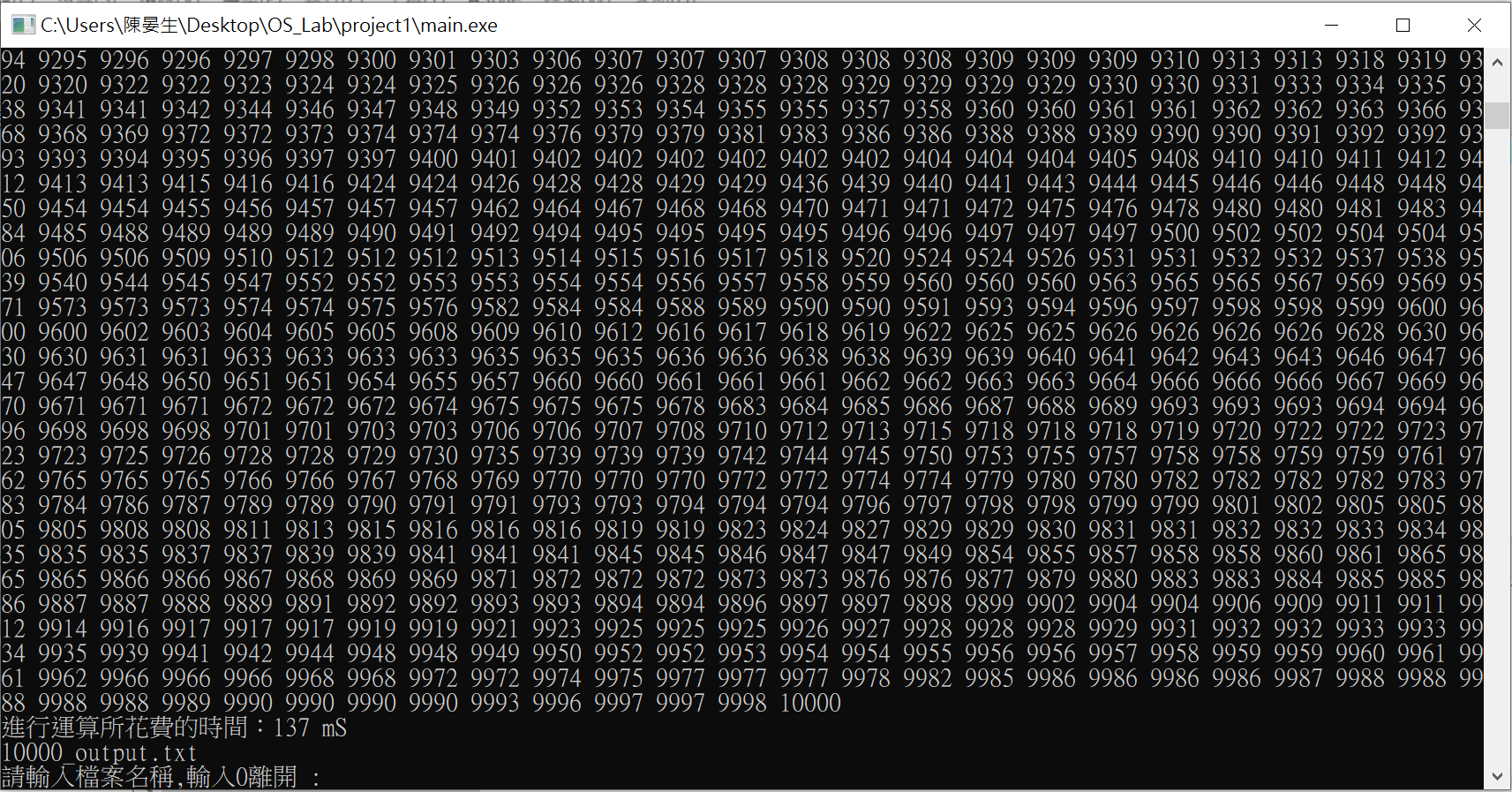
三種任務之中，第一題是單純的只用Bubble Sort去跑，泡沫排序是屬於較花時間的排序方法，因此資料筆數增加，處理時間呈現類似指數的飆升；第二和第四題的差別在於一個使用k個Thread進行Bubble Sort，一個用Process，由結果可以看出，同樣切成K份進行Bubble Sort並且進行k-1次的Merge Sort，使用執行序的結果比起使用Process快上非常多。

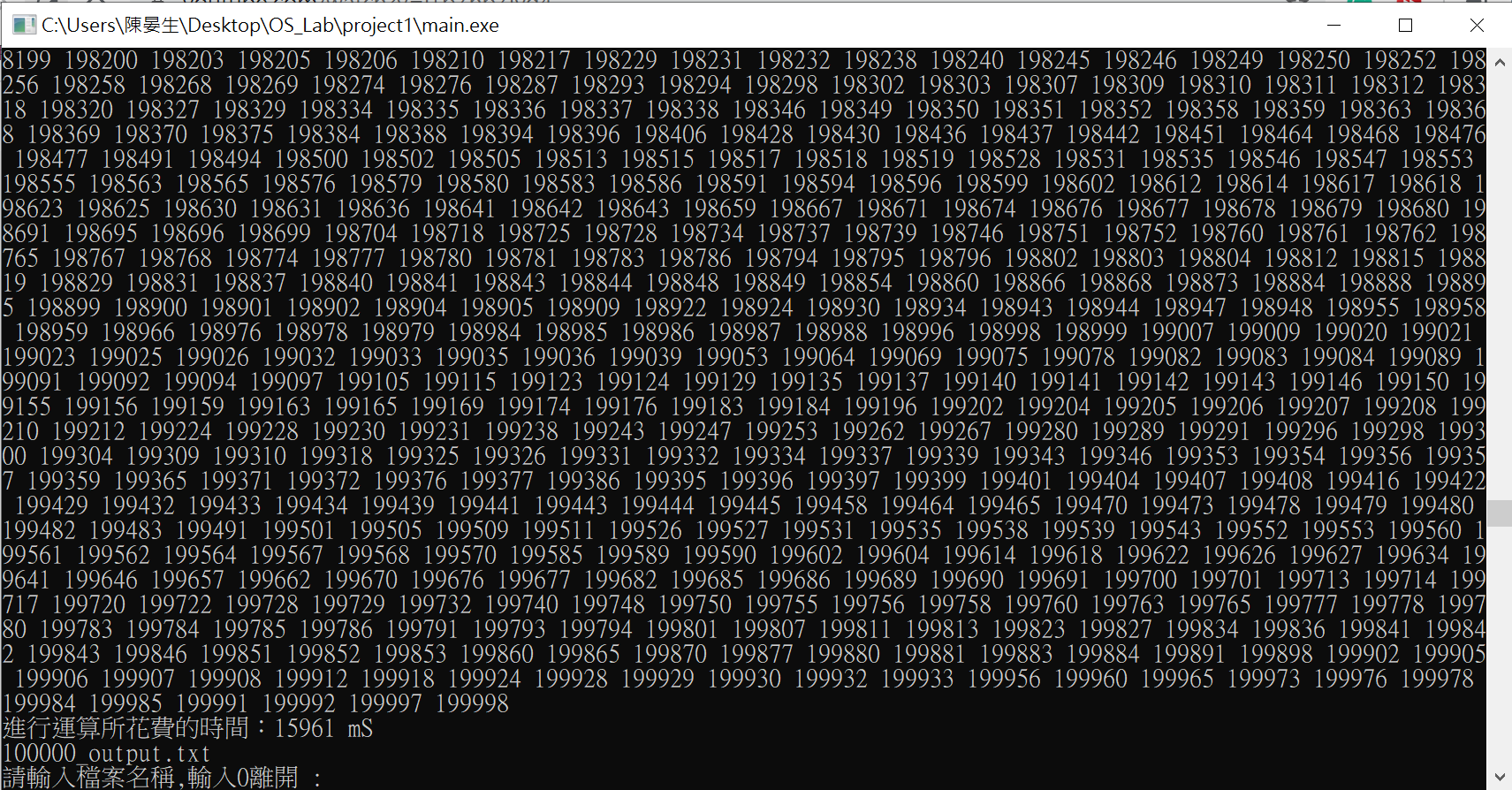
結果圖表:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 第一題 | 第二題 | 第四題 |
| 一萬筆 | 137 | 6 | 57 |
| 十萬筆 | 15961 | 142 | 1446 |
| 五十萬筆 | 376844 | 3426 | 22237 |
| 一百萬筆 | 1493880 | 13784 | 86352 |

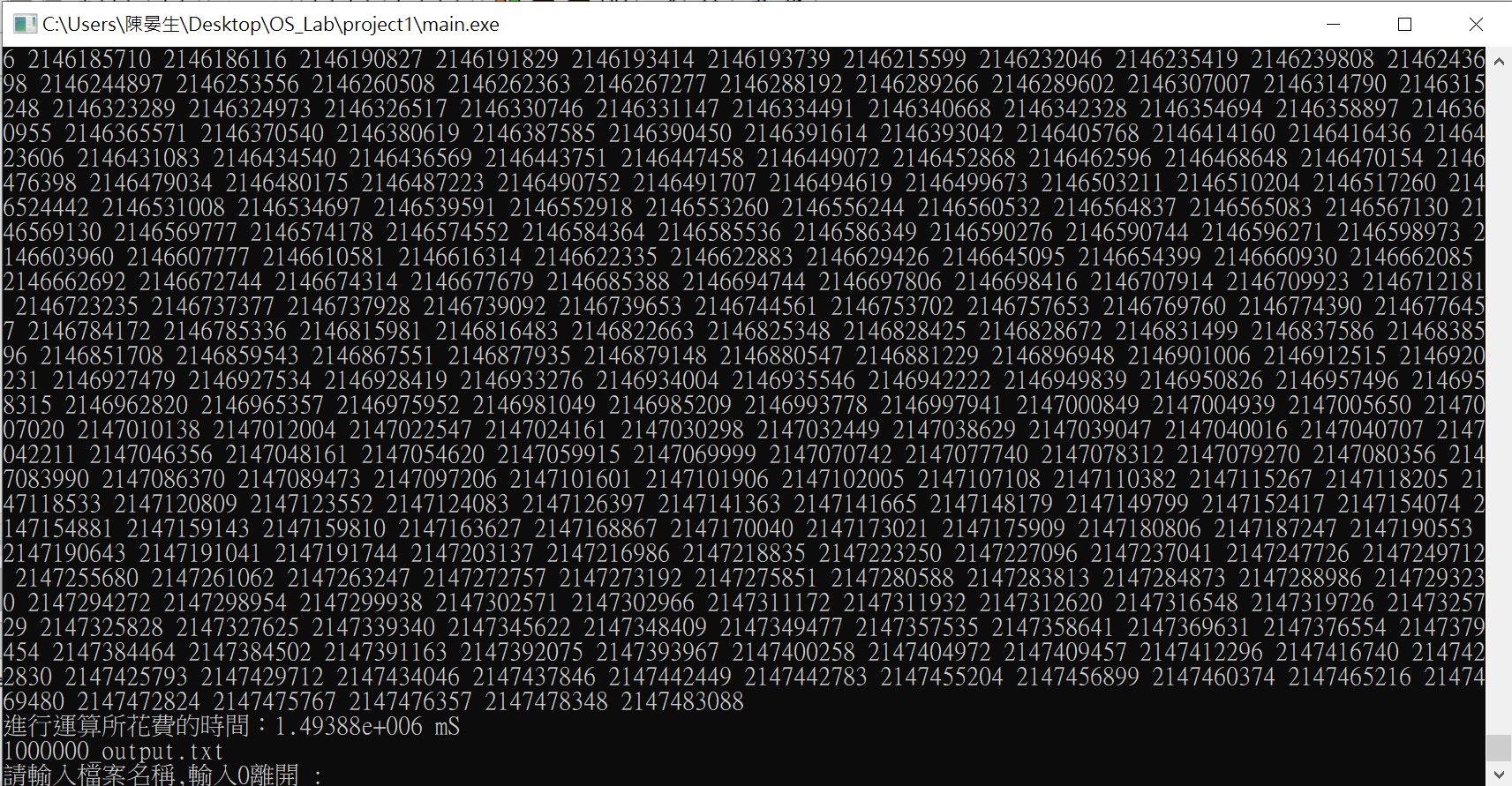
以下為各個任務的輸出結果截圖:

任務1(依序為一萬、十萬、五十萬、一百萬筆):

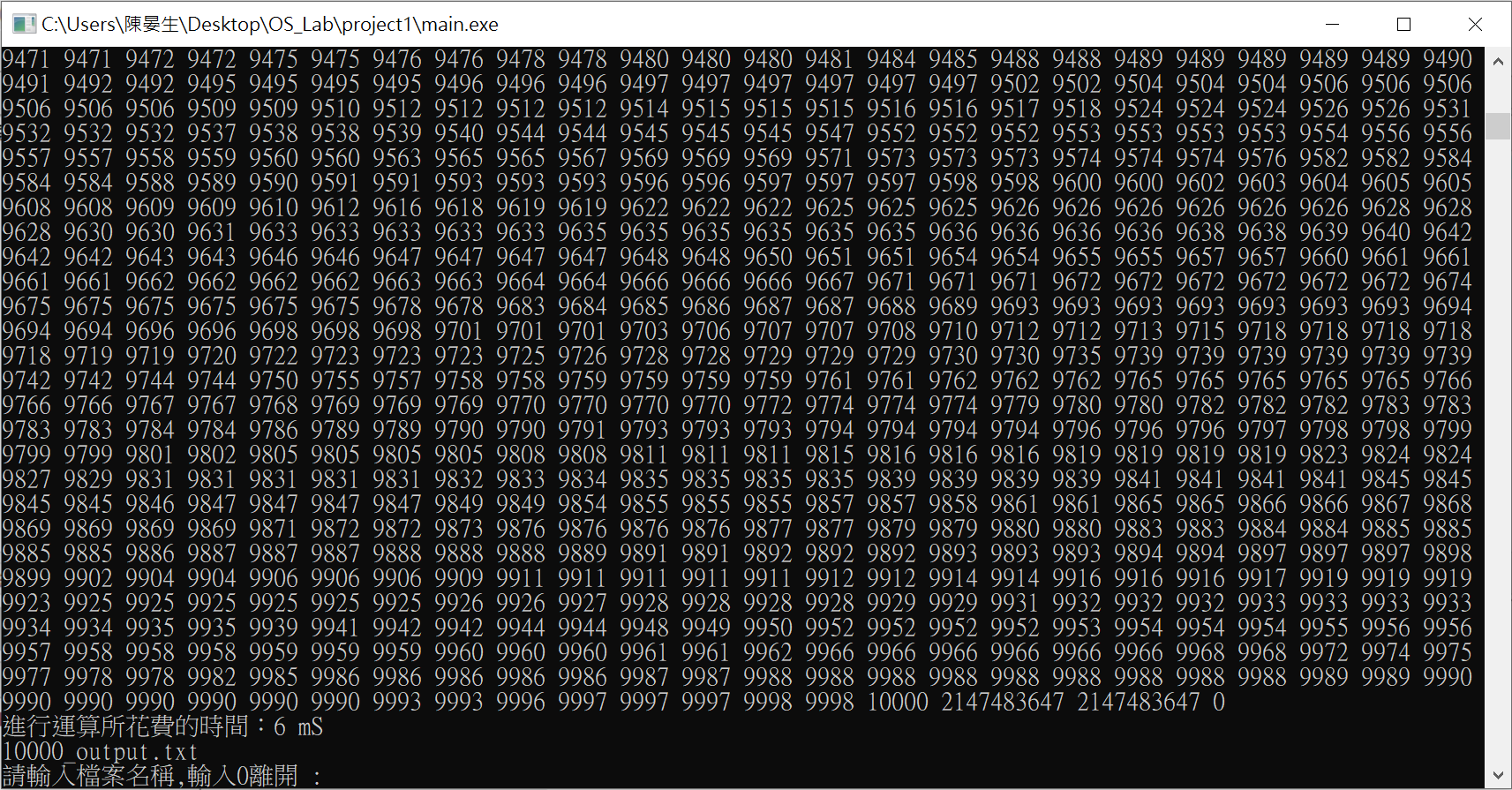


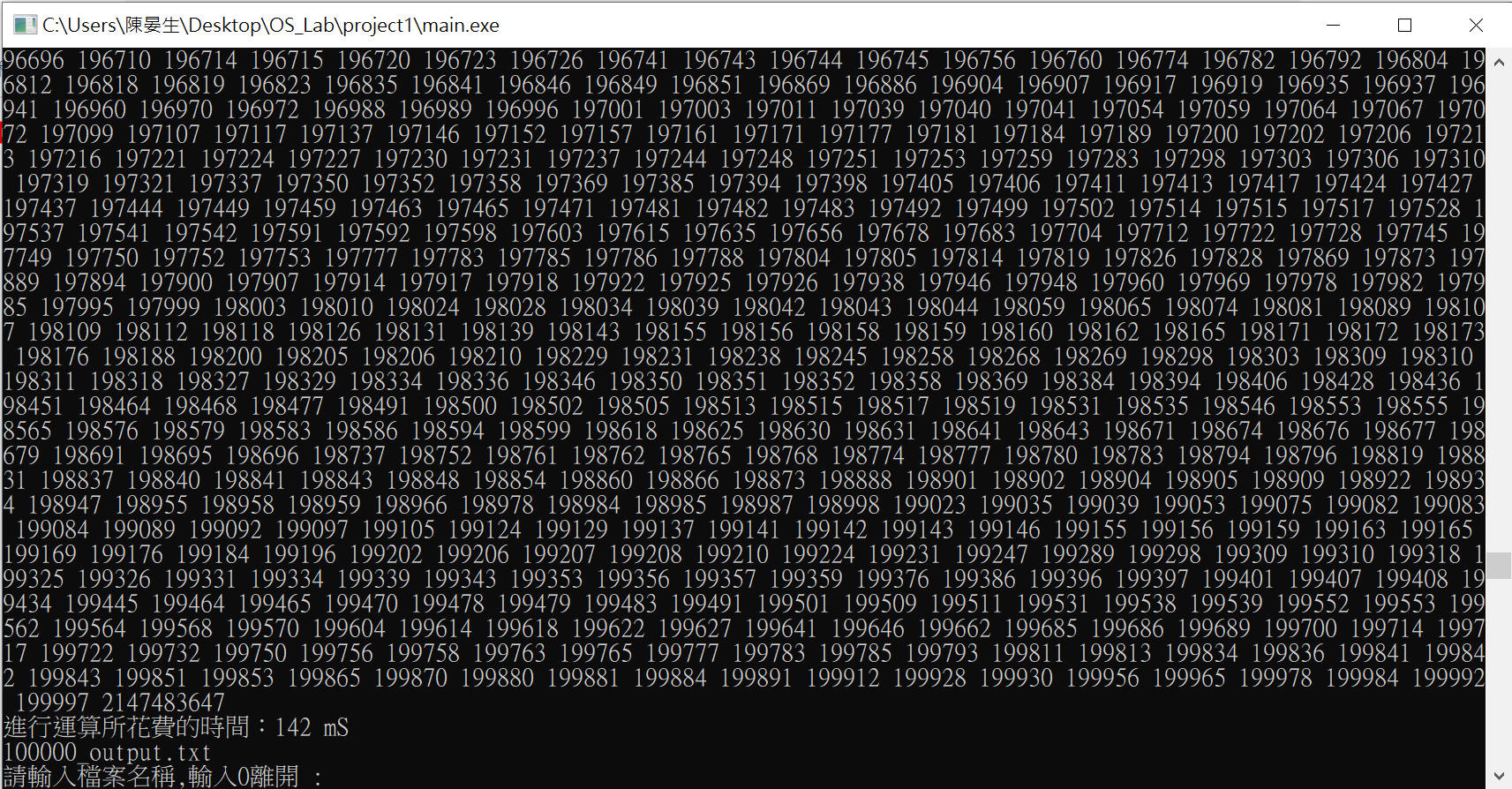


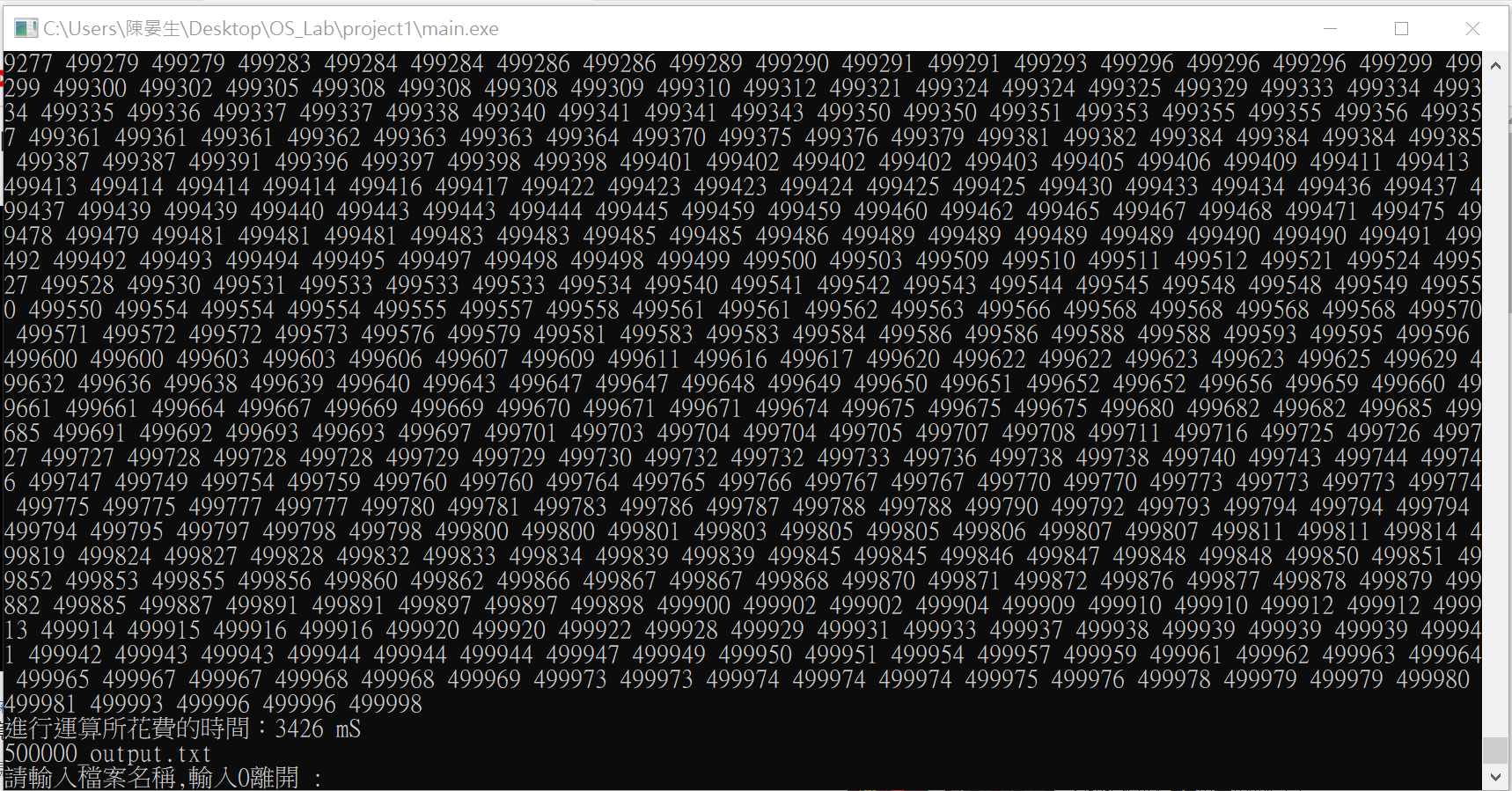


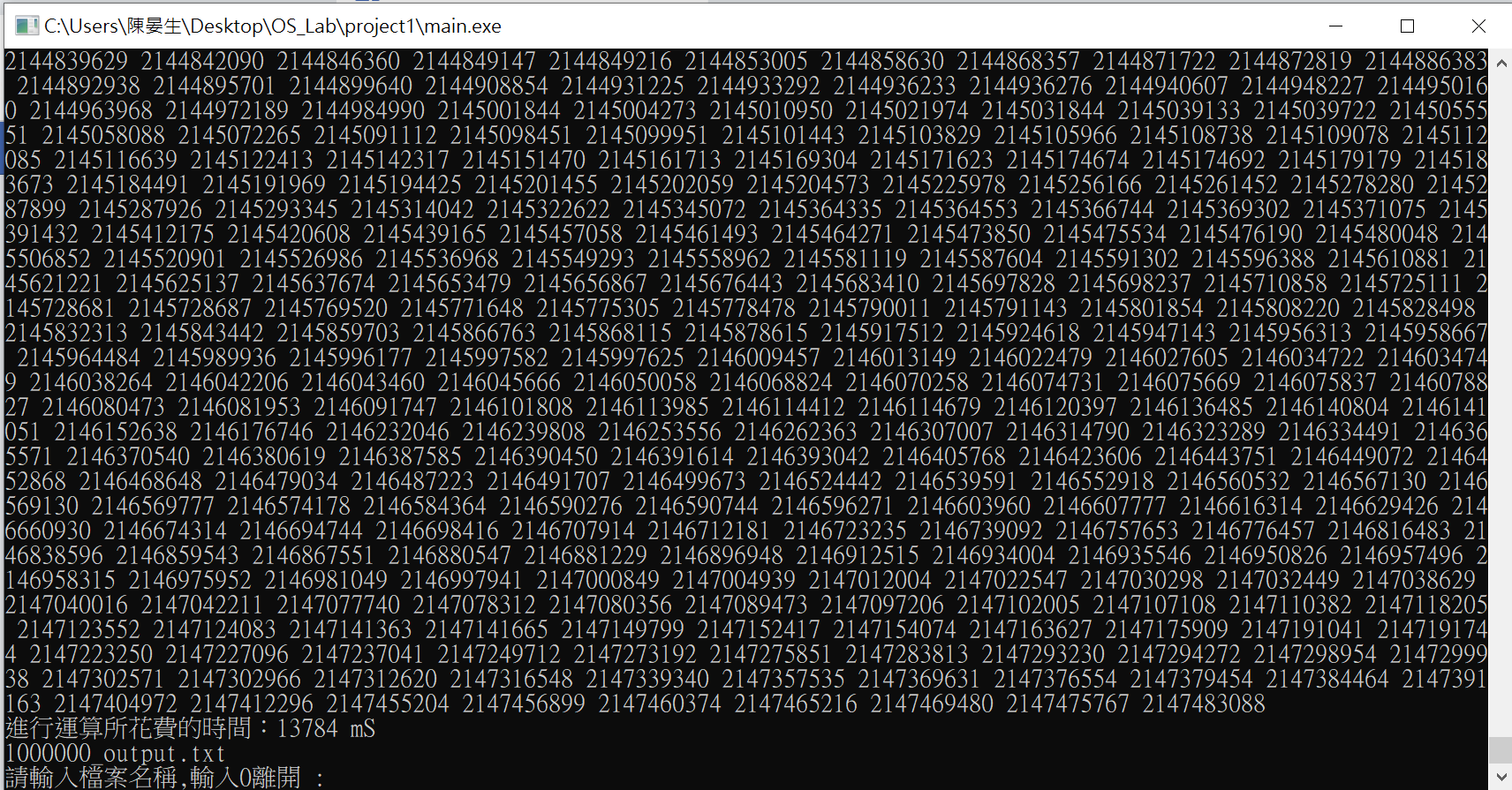


任務2(依序為一萬、十萬、五十萬、一百萬筆):

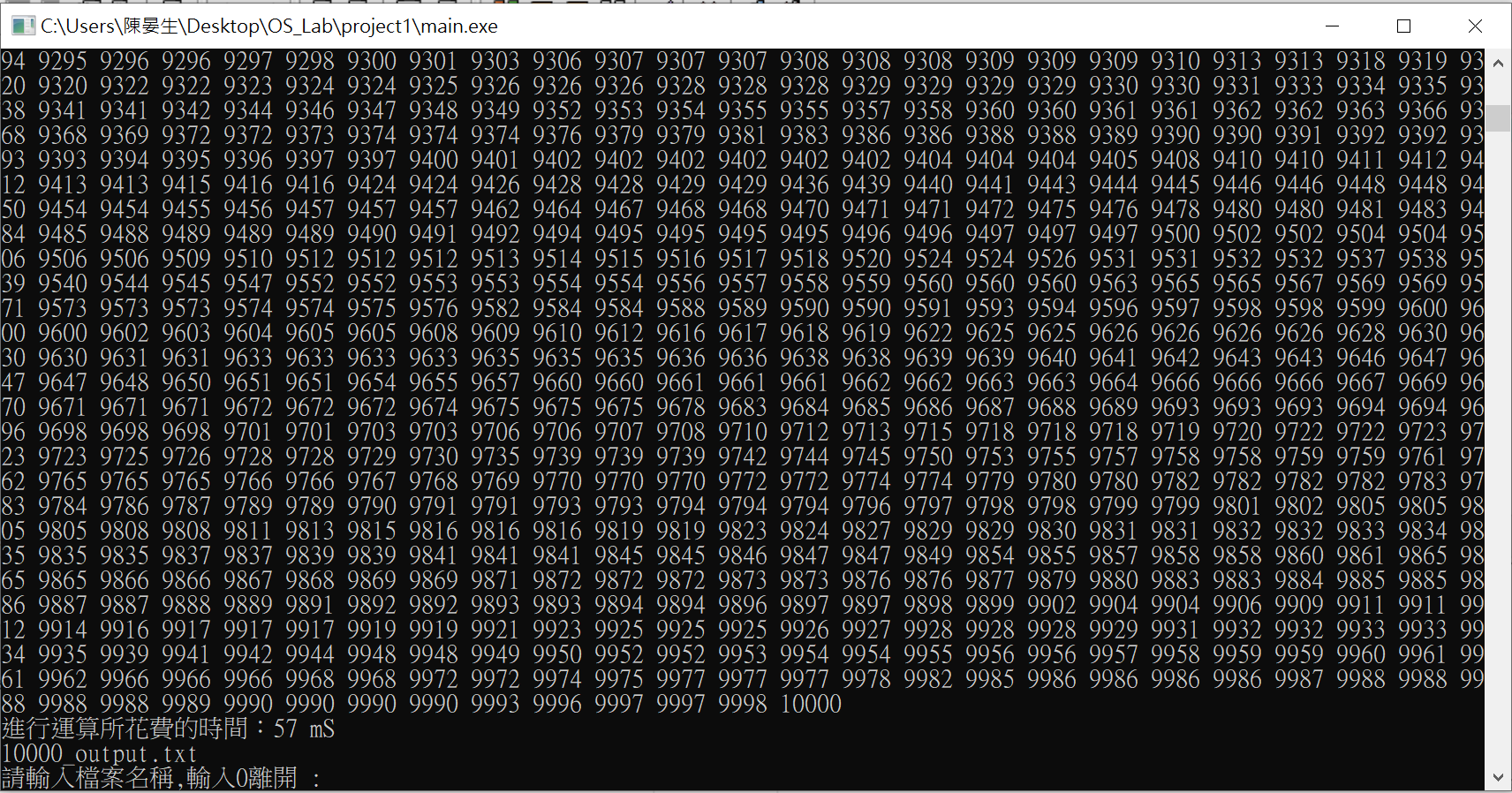


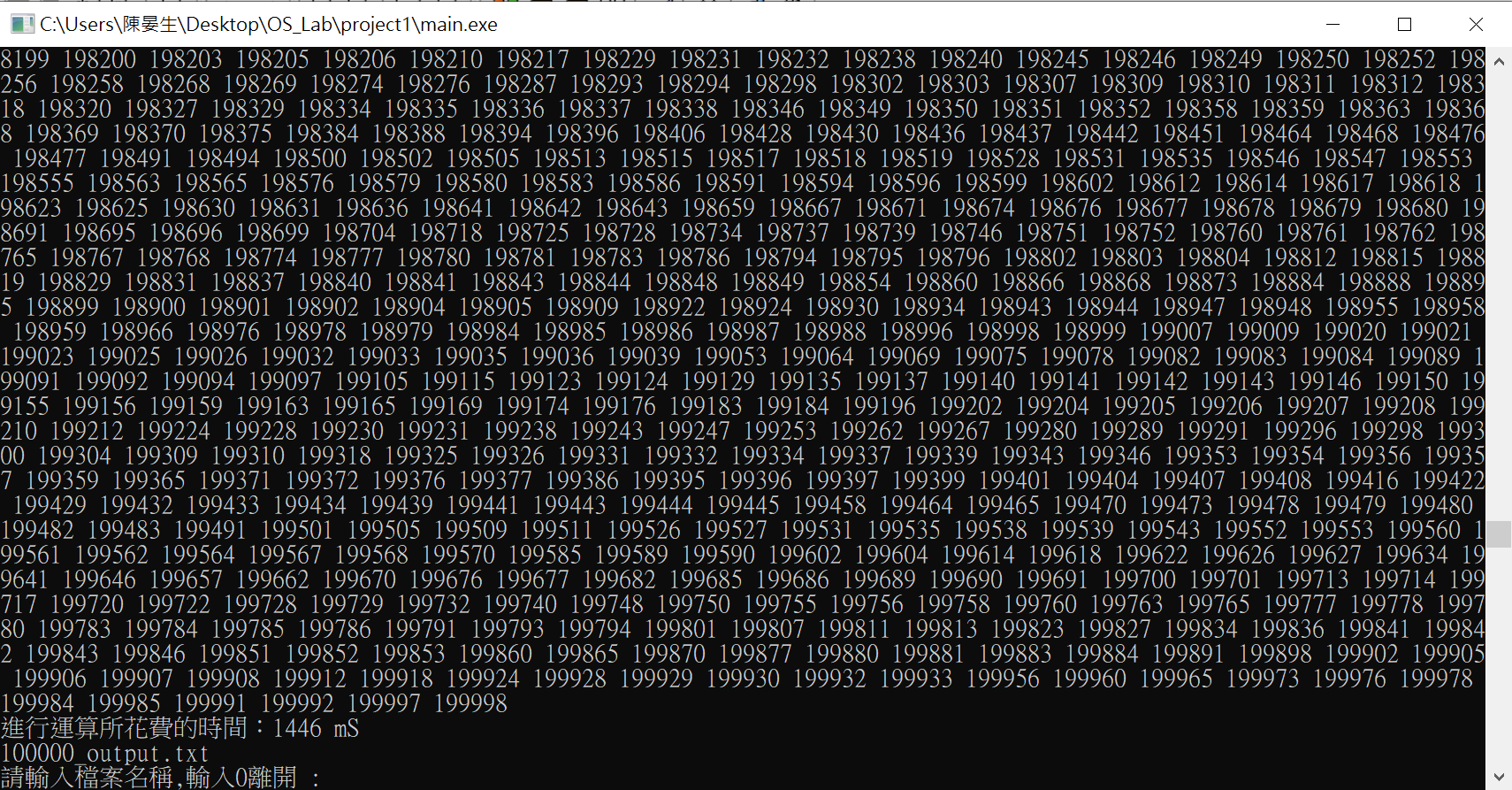


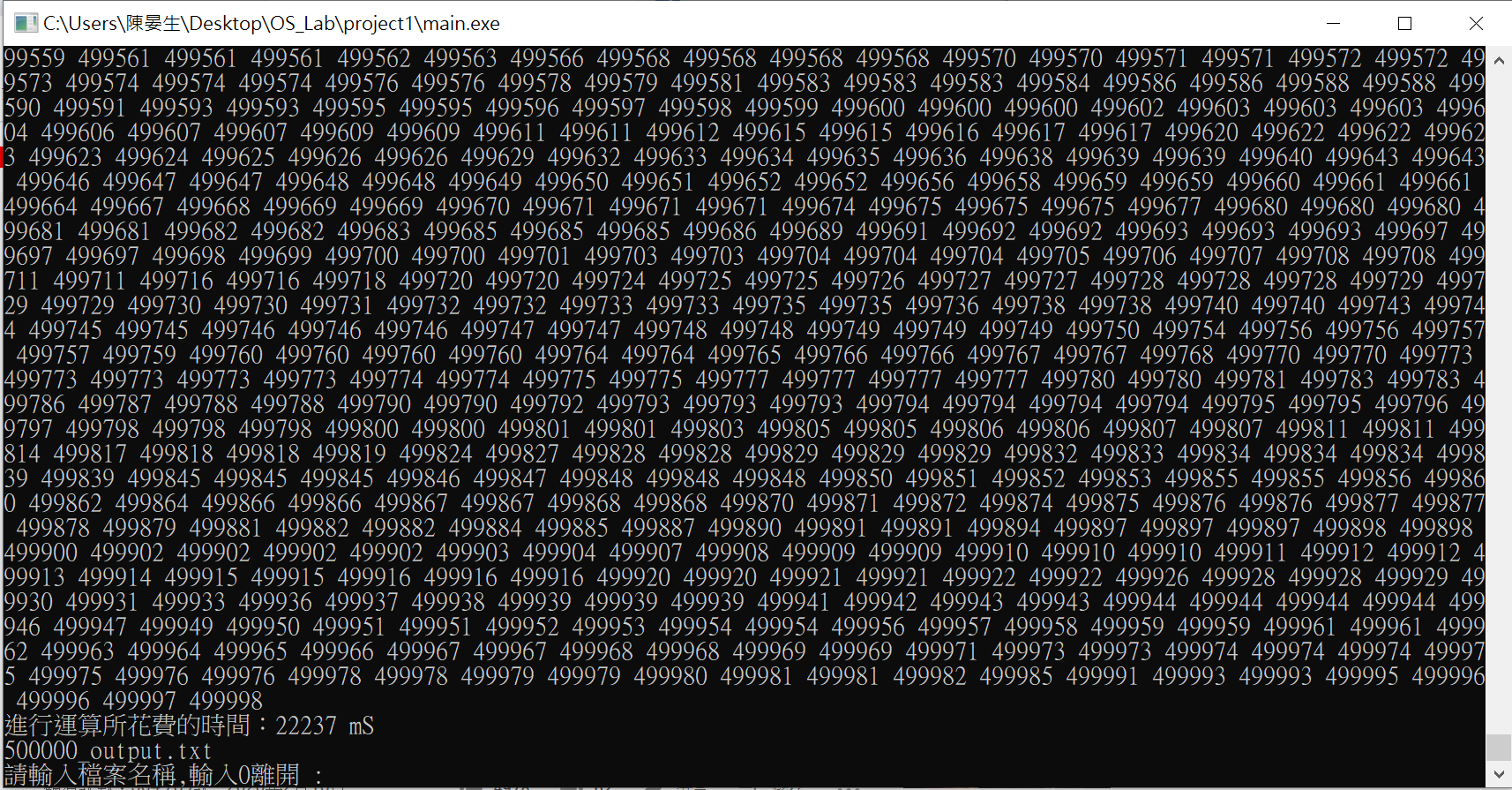


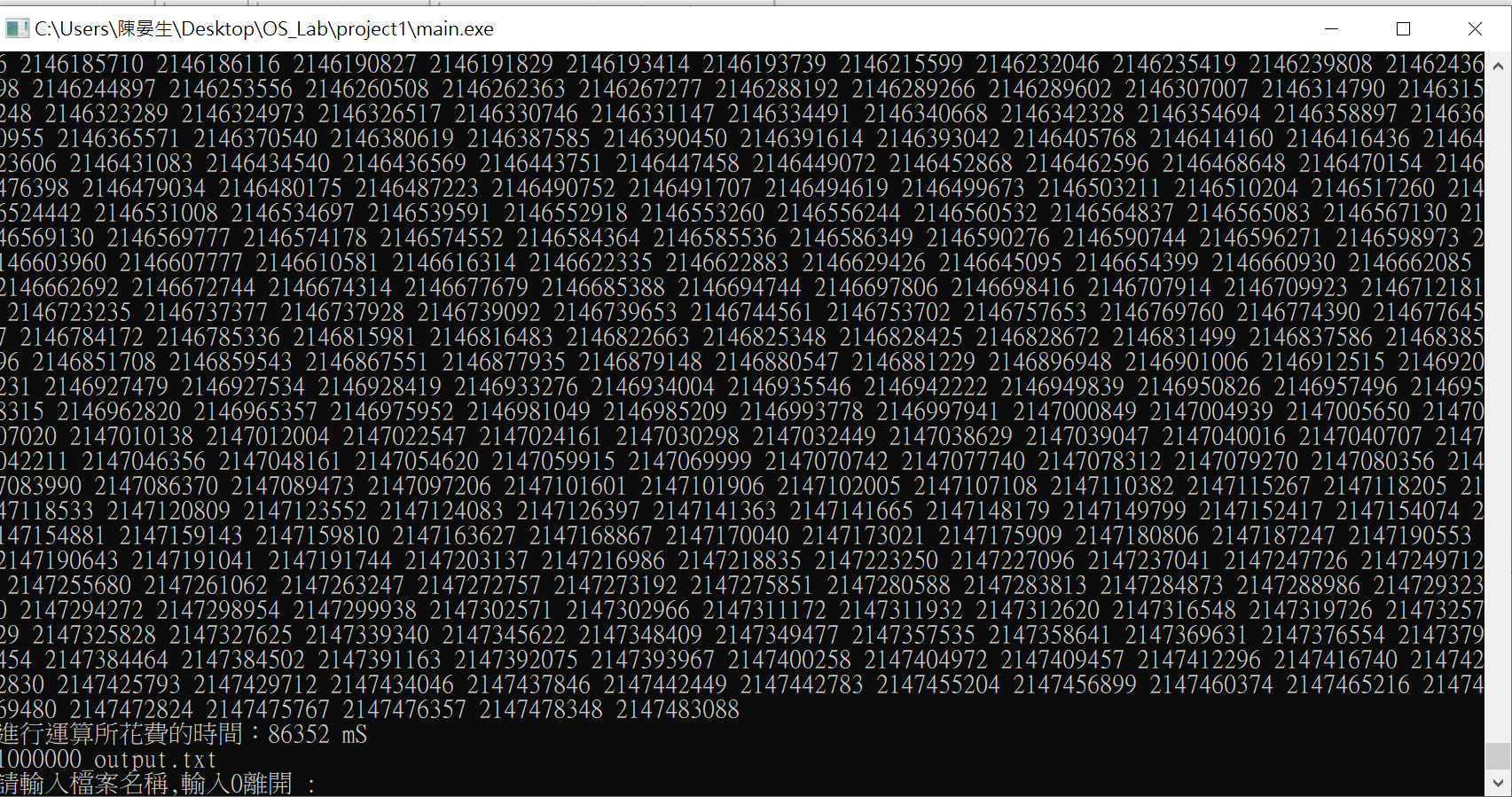


任務4(依序為一萬、十萬、五十萬、一百萬筆):









另外，任務三需要用到的Fork函數，因為在window系統中無法使用，所以筆需架設虛擬機器用Linux系統才能執行，但因為Dev C++不支援此功能，因此我有另外使用支援連線至虛擬機功能的Visual Studio執行程式，但資料筆數50萬以上的兩個檔案因未執行時間過久電腦沒辦法負荷跑不出結果。但因為無法成功地架設虛擬機，因此第三題我沒有完成。

以下為一些Visual Studio執行結果:

