**Problem 1 :**

**For regular function calls :**

**1st run: 33.965 seconds**

**2st run: 29.477 seconds**

**3st run: 30.273 seconds**

**4st run: 26.151 seconds**

**5st run: 33.624 seconds**

**6st run: 28.305 seconds**

**7st run: 27.412 seconds**

**8st run: 28.656 seconds**

**9st run: 28.601 seconds**

**10st run: 29.3 seconds**

**Avg\_inline = 29.5764 seconds**

**For inline usage :**

**1st run: 30.407 seconds**

**2st run: 31.176 seconds**

**3st run: 31.65 seconds**

**4st run: 31.66 seconds**

**5st run: 29.934 seconds**

**6st run: 28.179 seconds**

**7st run: 27.799 seconds**

**8st run: 26.426 seconds**

**9st run: 27.549 seconds**

**10st run: 27.378 seconds**

**Avg\_no\_inline = 29.2158 seconds**

**Observation :**

**從上述數據的顯示結果，可以看出其使用inline和使用regular function calls相比而言，inline的平均時間要比regular function calls還要來的少了0.3606秒，從這裡可以得出在相對於重覆呼叫函式而言，沒有比直接複製到主程式當中來執行還要來的快，可是像這一種的效果主要建立於當其函式的程式碼比較短並且又一直不斷的重覆呼叫時，才有辦法讓其執行的時間還要來的少，可是當其要執行函數內部的程式碼所需的時間大於處理呼叫函數的時間時，則能節省的時間就比較少，更甚至不會產生任何節省時間的效果，正如教授當時在課堂上教inline以及助教在加強班提到inline的時候，都有特別的去告訴我們，其inline儘管加到了函式的前面，可是它並不是強制編譯器來去執行這個inline的函式，而是去建議編譯器來去執行這個inline的函式，所以編譯器它是會去做選擇的，它會去根據當下的運算所需花費的時間以及程式碼的長度等等的因素，來去決定是否來去執行inline或者是執行regular function calls。**

**而我在這一次的實驗當中我也注意到了一點並且也一直在思考，那就是儘管今天所試驗的程式碼裡頭所使用的函式的程式碼確實是很短沒錯，可是整體實驗出來的結果卻也只相差了0.3606秒，對於上頭的數據所顯示的秒數都平均分布在26秒至33秒之間，其實inline與regular function calls彼此的數據相比出來的結果並沒有相差太多，這也就導致了兩者的平均秒數差距就來的十分的微小，那我就一直在思考一點，就是會不會硬體對於其結果的影響更來的大呢?我也去看過其他同學所跑出來的數據結果並且也去分析每一台電腦的硬體效能，我發現當其硬體效能越高時，則其不論是有無使用inline，對於執行出來的結果幾乎是差不了多少，更甚至看不出任何的差距，可是對於硬體效能相對較差的電腦就可以發現到有無inline的結果的差距是十分的大。**

**所以由以上可得出，儘管相差的秒數十分的微小，可是並不能夠來去完全否定inline的實用度，畢竟一段程式碼當其運行在不同的平台、編譯器以及硬體效能等等，其執行出來的結果都是不相同的，有時有無inline也是有可能造成巨大的影響。**