* (p1~p2)老師及各位同學大家好，我們是第17組，這裡是我們的outine，分別會去介紹我們做的主題，使用的演算法，選擇的平行程式，在實驗過程的加速表現，以及過程中所發現的問題以及所得到的結論，最後是Ｑ＆Ａ
* (p3)首先，在電腦科學的領域裡，我們最一開始接觸的也是非常常見的資料處理通常都是排序，排序是相當常見且被運用很平凡的演算法，因此我們希望可以透過平行程式對於最常見的演算法來最加速，並且從過程中觀察資料量大小的加速效果以及不同的排序演算法中，所呈現效能間提升及下降的變化。
* (p4)因此我們在這麼多排序法中我們選用了Bubble,Merge,以及quick，這三種sorting Algorithm算是相當基礎也是廣泛被用到的排序法，借此機會，去觀察加速的效果
* (p5,p6,p7)首先大概敘述一下 這三種排序法的概念和執行步驟：
  + Bubble sort:
    - 相鄰的兩個元素做比較，若前者較大則交換位置
    - 持續進行比較，則最後一個元素就會是最大的值
  + Merge:
  + Quick:
* (p8)接著可以看到這張表格
  + 這是原來用serial的方式去執行，所得到的效能表現，分別以1萬筆資料，25萬筆資料以及50萬筆資料所得到的結果
* (p9)在這裡我們將用openMP去改寫程式希望達到加速
  + 選用的原因首先是openMp較早開始學習，對於使用和實作的把握度較高
  + 再來因為openMP不需大幅度的改寫程式碼，只需要在code上面加上annotation和微調程式碼即可
  + 也因為相當普遍且具移植性，在相當多平台都有相當多資源可了解
  + 在這裡我們是在gitHub上來一起完成這個專題
* (p10)Bubble sort在平行加速的過程，整理了以下數據我們可以看到資料量在10000，25萬，50萬時，都在4thread達到最高的加速,在thread4之後speedup就開始下降，甚至在執行時間也算是大幅度的增加
* (p11)Merge sort 的數據可以看到在thread 8 不管在資料量小或是資料量大時還是有很顯著的加速
* (p.12)quick sort從數據中可以看到說在1萬筆資料時,thread2的speedup是比較高的，在25萬筆以及50萬筆則是會在4thread才會到最高的speedup