1. 簡介

‧Main()

輸入可能的指令，若指令為’0’，程式執行結束；若指令為’1’，執行任務一；若指令為’2’，執行任務二；如果都不是，請使用者再輸入一次指令。

‧任務一

利用Readcom1()將input.txt檔讀入事件清單arrays，若寫入成功呼叫Shellsort()進行排序，主要概念是由大到小選定gap(最終gap=1)，比較第i筆與第I + gap筆資料的arrival，若前者「大於」後者，則交換前後資料；若前者「等於」後者，進階比較第i筆與第i + gap筆資料的id，若前者大於後者，則交換前後資料，每一回合重複執行至沒有資料可以交換為止，排序完成後匯出至sorted.txt檔，最後分別測量讀檔、排序和寫檔的時間並顯示於螢幕上。

‧任務二

利用Readcom2()將排序好的sorted.txt檔讀入事件清單arrays，接者Item cpu, queues[3]表示處理情形，將第一筆資料存入cpu，第二、三、四筆分別存入queues[3]，判斷if(cpu.timeout>=cpu.start+cpu.duration)，成立則CPU時間time= cpu.start+cpu.duration，並將資料存入完成清單，否則time=cpu.timeout，且將資料存入取消清單，第五筆資料開始檢查if(arriva<time)，成立則存入取消清單，當queue[]的timeout<time時，也將資料存入取消清單，將處理完的資料移除，其餘資料往前遞補，空的部分以arrays中資料取代，若資料讀完則不再放入，cpu的資料全部處理完成後，分別將完成清單及取消清單寫入output.txt檔，並計算平均延遲時間和成功率。

‧發現與心得

這次學習到的新觀念:佇列(queue)，佇列的特別之處在於FIFO(First in first out)，符合現實生活中排隊的情形，是常用的資料結構，而在實作佇列的部分學習到可以利用指標和陣列的方式，個人認為使用指標的方法能更具體的理解程式運行的概況。

1. 流程圖
2. 答問

‧插入排序法的概念是取出後半部未排序的一個值，插入在前半部已排序的適當位置，方法為將第i個值插入「前i−1個排序過」的陣列。

int i表示「目前要處理」的資料index(i)；

int j = i - 1用來指出前1 ~ i−1筆資料的index；

用一個for迴圈把每一個陣列元素都視為「目前要處理」的資料；

在for迴圈裡利用while迴圈把「目前要處理」的資料與前1 ~ i − 1筆資料做比較，找到適當的位置，將1 ~ i筆資料做排序。

‧比較

Shell Sort Insertion Sort

1. 2ms 11ms

2. 5ms 29ms

3. 9ms 48ms

Average 5.33ms 29.33ms

‧希爾排序法又稱作改良的插入排序法，顧名思義效率上會比插入排序法更快速，原因是因為希爾排序法在插入排序法的基礎上添加了gap的概念，讓原本的插入排序法可以分組執行，並且元素的移動距離可以大於一，使得序列後端比較小的元素不需要大老遠的從後端以一次一個的距離移動到前端。