設計：
FIFO:簡單的先進先出，用 qsort 將每個 process 依照準備時間排序。後開啟迴圈後每輪檢查符合準備時間的 process，並設立等同符合準備時間數量的變數。依該變數檢查可執行的 process，由於每個 process 可以執行到完不被搶先，故能先計算其預計結束時間，並依該時間進行 wait。

SJF：分別依據準備時間、執行時間 sort 出兩個 process 列表，再依據短時間者優先原則計算出固定的執行順序列表，有了該列表後執行方式與 FIFO 同。

PSJF:要先行計算執行列表得花許多時間故不計算。在執行迴圈中每輪判斷是否有 process 準備完成。並在準備完成中的 process 中判斷使否有執行時間更短者，用該 process 取代原本的 process，即將優先權分別設為偏高即偏低。每輪都該檢查是否有 process 完成執行。

RR:設一變數代表上次 context switch 的時間，若該變數大於設立的時間閥值(這裡用 500)則將順位排給下一個符合準備時間病還沒執行完成者

2.結果

FIFO1:
P1:39153
P2:39162
P3:39167
P4:39168

FIFO3:
P1:39440
P2:39488
P3:39513
P4:39530
P5:39532
P6:39541
P7:39553

SJF3:
P1:39891
P4:39904
P5:39905

P6:39906
P7:39925
P2:39947
P3:39972
P8:40010

3. 比較實際結果與理論結果，並解釋造成差異的原因

很遺憾的，到目前為止，每項距離理論值都有些許差距，FIFO 因為我們沒有用到為了這個所特別製作的 function SCHED_FIFO 可能會有點差異，這樣的情況在其他的 input 也可以發現。

第二，我們沒有最小化 scheduler and child processes 的時間差，也沒特別設置 waiting processes to IDLE，這都可能導致時間的延長。

第三，不知是否為版本問題，我們的 RR 和 PSJF 無法在我們三人的電腦上實際 run 出來，但可以在 204 之類的地方進行實測，也遇到了可在我們電腦上跑的 SJF 與 FIFO 一樣的問題，可以說是待解決的問題

第四:只有用個別測資來操作

4.分工

K
SJF: b05902107,b05902123(輔
FIFO: b05902107bb04902067(輔
PSJF:b05902123,b05902107(輔
RR:b04902067,b05902123(輔