OS HW3 Report

Name: 莊婕妤
Student ID: 109550182

Question	Answer
Q1. Briefly describe your data structure for recording process' time or anything you need to record.	我創了一個叫 process 的 data structure， 結構如下所示： PID : 記錄我讀取這些 process data 的順序 arrivalTime : 記錄他的 arrival time burstTime : 記錄他的 burst time remainTime : 記錄他還剩下多少時間才會跑完 waitTime : 記錄他的 waiting time turnAroundTime : 記錄他的 turnaround time waiting : 記錄他有沒有在 round robin 的 ready queue 中（實作 SRJF 時不會用到） finished : 記錄這個 process 是否跑完

```
struct process {
    int pid = 0;
    int arrivalTime = 0;
    int burstTime = 0;
    int remainTime = 0;
    int waitTime = 0;
    int turnAroundTime = 0;
    bool waiting = false;
    bool finished = false;
};
```

Q2. How to simulate process scheduling?	我宣告了一個 integer 變數 current_time 來模擬時間軸，並用一個 while loop，在每個時間點判斷哪一個 process 該先執行。若所有 process 皆執行完成，則 while loop 會終止。 判斷哪一個 process 先執行的依據會根據 scheduling method 而有所不同： SRJF : 我讓他依照 remaining time 的大小去做 sorting，剩下的時間越少，會排在越前面 Round Robin : 這邊多設了一個變數 time quantum 來讓各個 process 輪流執行 Multilevel Feedback Queue : 這邊多判斷了在該時間點要執行哪一個 queue 中的 process
Q3. Some problems you meet and how to resolve them.	這次作業我卡關最久的地方在於實作 round robin 時，一直進入無窮迴圈。後來才發現，當我把 list[ready_queue.front( )] 存成 chosen 實作時，我並沒有改到 list[ready_queue.front( )] 裡面的值，而是一直在對他的 copy 做事情。將 chosen 改成 pointer 實作後，就一切正常了。 另一個我遇到的問題是，如何處理 HW3 Spec 中第 7 頁提及的 case。最後我多使用了一個變數 previous 來存上一次執行的 process index，然後在遇到該狀況時讓他排在 queue 的最後，讓後面的 process 先執行。 在實作 Multilevel Feedback Queue 時，我一開始也不知道怎麼讓 Q1 在被 Q0 的 process 中斷後，讓原本在執行的 process 排到 queue1 的後方。後來我也是宣告了變數 previous，記錄上一次跑的 queue 為 Q0 或 Q1，若符合上述條件時，將原本排在 queue1 最前面 process 的往後挪。

Q4. What you learned from doing OS hw3 and something you want to discuss with TAs.	在準備期中考時，我大概熟悉了許多 process scheduling methods 的運作原理，但僅限於自己用紙筆去推算誰先誰後。這次的 HW3 帶我用另一個更有系統的角度去認識這些演算法，也讓我更加了解如何用 c++ 去實作這些 process scheduling methods。