// 只上傳可成功編譯的原始碼(.cpp/.c/.h/.hpp)含註解、檔名請用「DS1ex6_分組編號_學號」，欠缺任何一項各扣 5 分！
// 程式碼前三行必須要有註解附上該組每位同學的中文姓名和學號，資訊不完整先扣 5 分！
// 每組只需要一位組員上傳程式碼和貼文，務必要在標題加上分組編號，兩份結果或標題不正確各扣 5 分！
// 非 C/C++程式 或 無法成功執行 一律視為「未完成」並以零分計！

一、題目

完成下列兩項任務，將二者整合在一個簡易選單下，未整合或介面無法連續執行先扣 5 分。

前言：

- **假設 CPU 排程處理的工作（job/process）表示成四個欄位：「工作編號」JobID、「進入時刻」Arrival、「工時長度」Duration 和「逾時時刻」TimeOut，Duration > 0 且 Arrival + Duration <= TimeOut。**
- **資料檔第一列是四個欄位名稱，其餘每一列代表一筆工作，四個欄位的數值均為正整數，以定位符號（'/t'）間隔，依「進入時刻」遞增排序，相同者則以「工作編號」遞增排序，檔名如 input601.txt。**

佇列模擬原則：（違反一項扣 10 分）

1. **每一筆工作一旦開始執行後都不能被中斷（not preemptive），進出佇列一律採用先進先出（FIFO）策略。**
2. **佇列必須實作成 C++ Class，雙 CPU 分別編號為 1 號和 2 號，每個 CPU 的佇列空間上限都只能存放最多 3 筆工作，並各有一個「可用時刻」，均預設為 0。**
3. **每筆工作可因佇列空間不足而立即捨棄，並累計於『捨棄工作清單』：欄位包括「工作編號」、「CPU 編號」、「捨棄時刻」Abort 和「延遲長度」Delay。**
   - 在上述情況下，「捨棄時刻」即為該筆工作的「進入時刻」，「延遲長度」則為 0。
4. **自佇列取出的工作，若檢查發現必然逾時，同樣也予以捨棄，並累計於『捨棄工作清單』。**
   - 在上述情況下，「捨棄時刻」為自佇列取出該工作的時刻，「延遲長度」則為「捨棄時刻」減去該筆工作的「進入時刻」。
5. **執行中舊工作的「完成時刻」和新工作的「進入時刻」相同時，一律先移除舊工作，以避免新工作因空間上限而無法進入。**

（任務一）最短佇列長度優先策略（Shortest Queue First, SQF）

定義：為每筆新工作選定一個 CPU 的 **SQF** 策略如下：

(1) 兩個 CPU 都是立即可用的（**「可用時刻」較早**且佇列是空的）：選 **1 號** CPU，立即執行或捨棄這一筆新工作。

(2) 只有一個 CPU 是立即可用的：選立即可用的 CPU，立即執行或捨棄這一筆新工作。

(3) 兩個 CPU 都不是立即可用的，且至少一個佇列並非全滿：選**佇列長度**（存放工作筆數）

最短的 CPU；若兩者等長，則選 **1 號** CPU，將新工作放入其佇列。

(4) 兩個 CPU 都不是立即可用的，而且兩個佇列全滿：立即捨棄新工作，『捨棄工作清單』的「CPU 編號」設定為 **0 號**。

輸入：讀入已排序的工作資料檔，存放於一個動態陣列中。

描述：(1) 從第一筆工作開始模擬雙 CPU 排程的等候及執行狀態，先比較新工作的「進入時刻」和各 CPU 的「可用時刻」，依序分別處理兩個佇列內已可執行的舊工作。

- 依照各 CPU 的「可用時刻」獨立處理，若早於或等於新工作的「進入時刻」，即表示 CPU 已閒置，可執行佇列內的舊工作。若舊工作必定逾時，直接予以捨棄。

(2) 接著為新工作以 **SQF 策略**選定一個 CPU，直接執行或以 FIFO 方式將其加入佇列。

(3) 所有工作都處理結束之後，仍要依序分別處理完兩個佇列內尚未執行的工作。

(4) 可成功執行的工作，依序累計於『執行工作清單』：欄位包括「工作編號」、「完成時刻」Departure 和「延遲長度」Delay。

- 在上述情況下，「完成時刻」為開始執行該筆工作的時刻加上該筆工作的「工時長度」，「延遲長度」則為開始執行該筆工作的時刻減去該筆工作的「進入時刻」

(5) 模擬完成後，統計『平均延遲長度』（捨棄工作也要算在內）及『成功執行比例』（成功執行工作筆數佔所有工作的百分比），一律取四捨五入至**小數點後兩位**的數值。

- 在上述情況下，『平均延遲長度』為每筆工作的「延遲長度」總和除以工作總筆數，『成功執行比例』為『執行工作清單』的工作筆數除以工作總筆數。

輸出：依序將『捨棄工作清單』、『執行工作清單』、『平均延遲長度』及『成功執行比例』寫成一個文字檔（檔名如 **SQF**601.txt）。

（任務二）最早可用佇列優先策略 EQF（Earliest Queue First, EQF）

定義：為每筆新工作選定一個 CPU 的 **EQF** 策略如下：

(1) 兩個 CPU 都是立即可用的：同 SQF。

(2) 只有一個 CPU 是立即可用的：同 SQF。

(3) 兩個 CPU 都不是立即可用的：不管佇列長度，選**「可用時刻」最早**的 CPU；若兩者同時，則選 **1 號** CPU，將新工作放入其佇列或立即捨棄。

輸入：同任務一。

描述：(1) 從第一筆工作開始模擬雙 CPU 排程的等候及執行狀態，先比較新工作的「進入時刻」和各 CPU 的「可用時刻」，依序分別處理兩個佇列內已可執行的舊工作。

- 依照各 CPU 的「可用時刻」獨立處理，若早於或等於新工作的「進入時刻」，即表示 CPU 已閒置，可執行佇列內的舊工作。若舊工作必定逾時，直接予以捨棄。

(2) 接著為新工作以 **EQF 策略**選定一個 CPU，直接執行或以 FIFO 方式將其加入佇列。

(3) 所有工作都處理結束之後，仍要依序分別處理完兩個佇列內尚未執行的工作。

(4) 可成功執行的工作，依序累計同任務一。

(5) 模擬完成後，統計同任務一。

輸出：同任務一（檔名如 **EQF**601.txt）。

二、參考範例

Input a file number (e.g., 601, 602, 603, ...): 601

JobID	Arrival	Duration	TimeOut
101	3	9	12
112	5	6	13
103	6	7	15
106	6	9	17
108	6	8	18
105	10	6	20
104	11	9	22

// 依序為「工作編號」、「進入時刻」、「工時長度」、「逾時時刻」

（任務一）最短佇列長度優先策略

[Abort Jobs] // 『捨棄工作清單』saved as **SQF**601.txt

	JobID	CPU	Abort	Delay	
					// 依序為「工作編號」、「CPU 編號」、「捨棄時刻」、「延遲長度」
[1]	106	2	11	5	// 因 11+9>17，發現必然逾時，故予以捨棄
[2]	103	1	12	6	// 因 12+7>15，發現必然逾時，故予以捨棄
[3]	108	1	12	6	// 因 12+8>18，發現必然逾時，故予以捨棄
[4]	104	2	17	6	// 因 17+9>22，發現必然逾時，故予以捨棄

[Jobs Done] // 『執行工作清單』

	JobID	CPU	Departure	Delay	
					// 依序為「工作編號」、「CPU 編號」、「完成時刻」、「延遲長度」
[1]	101	1	12	0	// 基於 SQF 定義(1)分派給 **1 號** CPU 執行
[2]	112	2	11	0	// 基於 SQF 定義(2)分派給 **2 號** CPU 執行
[3]	105	2	17	1	// 因 11+6<20，從 10 等到 11 執行

[Average Delay] // 『平均延遲長度』(5+6+6+6+0+0+1) / 7

3.43 ms

[Success Rate] // 『成功執行比例』100 * 3 / 7 %

42.86 %

（任務二）最早可用佇列優先策略

[Abort Jobs] // 『捨棄工作清單』saved as **EQF**601.txt

	JobID	CPU	Abort	Delay	
					// 依序為「工作編號」、「CPU 編號」、「捨棄時刻」、「延遲長度」
[1]	105	2	10	0	// 因 **2 號** CPU 的佇列已滿，故立即捨棄
[2]	103	2	11	5	// 因 11+7>15，發現必然逾時，故予以捨棄

[3] 106 2 11 5 // 因 11+9>17，發現必然逾時，故予以捨棄
[4] 108 2 11 5 // 因 11+8>18，發現必然逾時，故予以捨棄

[Jobs Done] // 『執行工作清單』
JobID CPU DepartureDelay // 依序為「工作編號」、「CPU 編號」、「完成時刻」、「延遲長度」
[1] 101 1 12 0 // 基於 EQF 定義(1)分派給 **1 號** CPU 執行
[2] 112 2 11 0 // 基於 EQF 定義(2)分派給 **2 號** CPU 執行
[3] 104 2 20 0 // 因 11+9<22，沒有延遲

[Average Delay] // 『平均延遲長度』(0+5+5+5+0+0+0) / 7
2.14 ms

[Success Rate] // 『成功執行比例』100 * 3 / 7 %
42.86 %

// 注意：程式跑不出正確結果，或未依規定格式輸出，均視同「未完成」，並以零分計！

三、預交（分數不打折）的必要條件

步驟 1. 同組兩人均有簽到。
步驟 2. 期限前完成一項任務，成功上傳程式碼後找助教或「已完成同學」展示正確結果。
步驟 3. 助教在「上機評分表」上勾選已完成預交。

// 注意：兩項任務在上機練習時完成者，可預約提前機測，機測前要先在討論版貼文！

四、程式簡介、流程圖及答問

截止期限前必須在本次上機練習的討論版張貼這一篇文章，否則成績歸零。
1. 簡介：以文字簡述程式主旨，假設，遇到的困難和解法，勿直接剪貼題目字句！
2. 流程圖：每項任務各一張流程圖，以附圖置於貼文之後！
3. **答問：分別從『平均延遲長度』和『成功執行比例』比較兩項任務所採用策略的優缺點，並嘗試描述一個可能更好的策略。**

五、機測程序及分數配置

步驟 1. 已完成提前機測、未上傳程式碼或未貼文者，均不列入機測名單。
步驟 2. 遵循公告名單的指定助教和機測時段到機房，遲到或缺席者視同放棄，一律零分。
步驟 3. 只限下載所上傳的程式碼，重新編譯後執行。經助教同意，只能用自己筆電機測者，

一律先扣 10 分。

步驟 4. 機測評分後，助教將根據是否完成預交予以打折。

步驟 5. 兩人一組時，一人負責機測一項任務，若只有一人機測，將會少一項任務的成績。

項目 1.（任務一）　40%

項目 2.（任務二）　40%

項目 3.　(1) 程式碼和註解易讀性、執行介面友善度　10%

(2) 程式簡介、流程圖及答問　10%

六、機測的評分方式

前兩個項目在機測現場評分，項目 3.則在機測之後由助教自行檢視

（階段一：實作）隨機施測 1-3 個不同輸入

1. 答案完全正確　得 30 分
2. 輸出結果只出現 1 筆錯誤的答案，依助教指示及時修正　得 25 分
3. 輸出結果出現多於 1 筆錯誤的答案，依助教指示及時修正　得 20 分
4. 未能依助教指示及時修正，但是助教認定已大部份完成　得 10 分
5. 其他　得 0 分

（階段二：原理）抽問程式相關的 1-3 個問題

1. 回答正確且能清楚解說程式碼　得到 10 分
2. 無法正確回答 1 個問題或無法清楚解說 1 行程式碼　得 5 分
3. 無法正確回答超過 1 個問題或無法清楚解說 1 行以上的程式碼　得 0 分

// 注意：成績公佈後才開始以軟體及人工比對程式碼相似度，由老師做最後裁定，相似度高於門檻的雙方都一律零分。

七、進階練習（僅供參考，不計分）

單 CPU 的佇列模擬原則：(違反一項扣 10 分)

1. 每一筆工作一旦開始執行後都不能被中斷（not preemptive）。
2. 佇列必須實作成 C++ Class，佇列空間上限都只能存放最多 5 筆工作。
3. 每筆工作可因佇列空間不足而立即捨棄，並累計於『捨棄工作清單』：欄位包括「工作編號」、「捨棄時刻」Abort 和「延遲長度」Delay。
   - 在上述情況下，「捨棄時刻」即為該筆工作的「進入時刻」，「延遲長度」則為 0。
4. 自佇列取出的工作，若檢查發現必然逾時，同樣也予以捨棄，並累計於『捨棄工作清單』。
   - 在上述情況下，「捨棄時刻」為自佇列取出該工作的時刻，「延遲長度」則為「捨棄時刻」減去該筆工作的「進入時刻」。
5. 執行中舊工作的「完成時刻」和新工作的「進入時刻」相同時，一律先移除舊工作，以避免新工作因空間上限而無法進入。

（任務一）逾時工作刪減機制

定義： 先檢查新工作是否必然逾時，若是就立即捨棄，否則以先進先出策略加入佇列。若佇列已滿，先依序檢查佇列內的舊工作是否必然逾時，刪減後才將新工作加入佇列。

輸入：讀入已排序的工作資料檔，存放於一個動態陣列中。

描述：(1) 從第一筆開始模擬工作進入單一 CPU 排程的等候及執行狀態，未立即執行的工作先檢查是否必然逾時，若是就立即捨棄，並累計於『捨棄工作清單』，否則以**先進先出 (FIFO)** 策略加入佇列。

(2) 佇列已滿時，先執行**逾時工作刪減**(**E**xpired **J**ob **P**runing)機制，刪減的逾時工作也依序累計於『捨棄工作清單』，之後再依循上機練習五任務二的步驟進行。

- 在上述情況下，舊工作的「捨棄時刻」為新工作的「進入時刻」，「延遲長度」則為「捨棄時刻」減去該筆舊工作的「進入時刻」。

(3) 可成功執行的工作，依序累計於『執行工作清單』：欄位包括「工作編號」、「完成時刻」Departure 和「延遲長度」Delay。

- 在上述情況下，「完成時刻」為開始執行該筆工作的時刻加上該筆工作的「工時長度」，「延遲長度」則為開始執行該筆工作的時刻減去該筆工作的「進入時刻」。

(4) 模擬完成後，統計『平均延遲長度』（捨棄工作也要算在內）及『成功執行比例』（成功執行工作筆數佔所有工作的百分比），一律取四捨五入至**小數點後兩位**的數值。

- 在上述情況下，『平均延遲長度』為每筆工作的「延遲長度」總和除以工作總筆數，『成功執行比例』為『執行工作清單』的工作筆數除以工作總筆數。

輸出：依序將『捨棄工作清單』、『執行工作清單』、『平均延遲長度』及『成功執行比例』寫成一個文字檔（檔名如 **EJP**601.txt）。

（任務二）工時最短優先策略

定義：「工時長度」較短的工作優先執行，「工時長度」相等時，則以「進入時刻」比較早的優先，再相同者則以「工作編號」比較小的優先。

輸入：同任務一。

描述:(1) 從第一筆工作開始模擬單 CPU 排程的等候及執行狀態,未執行的工作先放入佇列。若超過佇列剩餘空間，工時較長者要被立即捨棄，並累計於『捨棄工作清單』。

(2) 佇列內的工作採用**工時最短優先** (**S**hortest **J**ob **F**irst)機制，佇列已滿時，若是新工作的「工時長度」比較短方可放入佇列內，原佇列內「工時長度」最長的舊工作則必須被捨棄，並累計於『捨棄工作清單』。

- 在上述情況下，舊工作的「捨棄時刻」為新工作的「進入時刻」,「延遲長度」則為「捨棄時刻」減去該筆舊工作的「進入時刻」。

(3) 可成功執行的工作，依序累計同任務一。

(4) 模擬完成後，統計同任務一。

輸出：同任務一（檔名如 **SJF**601.txt）。