1. 開發環境 :

作業系統 : windows

使用語言 : C++

測試環境 : Dev C++

1. 程式設計 :

Data structure :

Struct job{ // 紀錄job 的資訊

int n ; // 記錄此工作在原本的jobList中是第幾個，因為某些排 班器會將工作丟進queue中，所以要記住它是jobList 中的哪一個，才能修改資訊。

int ID ; // 工作的ID

int cpu\_burst ; // 工作的 CPU burst

int arrival ; // 工作的 arrival time

int priority ; // 工作的 priority

int waiting ; // 工作的 waiting time (每個排班器都將結果會回傳 成各自的job list ，所以會記錄各自的waiting 、 turnaround time)

int turnaround ; // 工作的 turnaround time

};

main :

vector<job> jobList ; // 存放讀入的所有工作的

int time\_slice, mode ; // input的 time slice 和mode

vector< vector<job> > result ; // 排班器輸出的jobList，有紀錄各自的waiting、turnaround time，mode 6會有六個result，其他則是一個

vector< vector<int> > workHis ;// 排班器執行工作的紀錄，會將執行的工作ID存入，若無工作執行則會存入-1

每個排班器有各自的class，先進行讀檔將資料存入jobList中，會先將工作依照arrival time 進行重新排序，再依照讀到的mode 呼叫對應的排班器進行排班，取得存有工作紀錄的workHis 和 result，最後再印出結果。

FCFS :

int time ; // 目前的時間

vector<int> workHis ; // 工作紀錄

當time >= jobList中第n個工作的arrival時，執行該工作直到工作結 束，並記錄至workHis中，並記錄該工作的waiting、turnaround time，直到所有工作都執行完，回傳有更新waiting、turnaround time的jobList。

RR:

int time ; // 目前的時間

int time\_slice ;

vector<int> workHis ; // 工作紀錄

vector<job> queue ; // 已到達的工作

將已到達的工作丟入queue中，執行queue中的第一個工作，若time slice 耗盡，就將工作放到queue的最後面，time slice重製，繼續執行 queue中的第一個工作；若目前的工作完成，就將工作移出queue 中，繼續執行queue中的第一個工作。每當更換工作時更新jobList中 將要執行的工作的waiting time，當工作完成時，紀錄turnaround time，當所有的工作都到達且queue中的工作都完成後回傳結果。

SRTF:

int time ; // 目前的時間

int time\_slice ;

vector<int> workHis ; // 工作紀錄

vector<job> queue ; // 已到達的工作

將到達的工作使用Insert() 根據CPU burst插入至queue中，執行 queue中的第一個工作，更新queue中其他工作的waiting，當有工作 完成時，紀錄它的turaround，並從queue中移除，繼續執行queue中 的第一個工作，直到所有的工作都到達且queue中的工作都完成後回 傳結果。

PPRR:

int time ; // 目前的時間

int time\_slice ;

vector<int> workHis ; // 工作紀錄

vector<job> queue ; // 已到達的工作

job nowJob ; // 目前在執行的工作

bool haveJob ; // 目前是否有工作在執行

將到達的工作使用Insert() 根據priority插入至queue中，若queue中 有工作，且目前沒工作在執行(haveJob == false)，將nowJob 設為 queue中的第一個工作，並將此工作從queue中移除，若目前有工作 但queue中的第一個工作的優先度高於目前的工作，就將nowJob 插 入至queue中，nowJob更換為queue[0]。當工作完成時紀錄 turnaround，將haveJob設為false；當time slice耗盡時，將nowJob插 入至queue中，將haveJob設為false。

當所有工作都到達、queue中沒有工作且沒有工作在執行，救回傳結 果。

HRRN:

int time ; // 目前的時間

int time\_slice ;

vector<int> workHis ; // 工作紀錄

job nowJob ; // 目前在執行的工作

vector<job> queue ; // 已到達的工作

將已到達的工作丟入queue中，用GetHR(time) 取得目前queue中response ratio最高的工作，將此工作設為nowJob，並將此工作從queue中移除，執行nowJob直到此工作完成，設定waiting、turnaround，在用相同方式取下一個工作執行，直到queue中所有工作都完成，在回傳結果。