1. 設計：

FIFO 的演算法：

先將全部的 process 按照 ready time 從小到大 sort 一次，然後按照這個順序給予大到小的 priority 後，丟進 priority queue 裡面，從 parent process 開始一個一個 fork 出 child，再 call exec。

SJF 的演算法：

先將全部的 process 按照 ready time 從小到大 sort 一次，如果有相同的 ready time 則 execute time 較小的排前面，同時將每個 process 的 ready time 和 execute time 加起來命名為 total time，從已經排序好的 process 中取第一個的 total time 往後尋找，只要在第一個 process 之後有找到 ready time 比當前 total time 還小的，則從該筆 process 之後便以 execute time 排序，直到排到有一筆 process 的 ready time 大於 total time，重複這個步驟直到排完

```
SJF(process_list){
    sort(process_list)
    for (p in process_list){
        p.total_time = p.ready_time + p.execute_time
    }
    c = 0;
    while(k < process_list.size()){
        if(process_list[c].total_time > process_list[k].ready_time)
            sort_by_exec(process_list[c+1:k])
            c = c + 1
        k++
    }
}
```

PSJF 的演算法：

先將全部的 process 按照 ready time 從小到大 sort 一次，如果有相同的 ready time 則 execute time 較小的排前面，接著按照 execute time 給予 process 權重，execute time 越低的權重越高，接著只要一有 process 的 ready time 結束進入到 priority queue 裡時就會按照權重高低重新 sort 一次，然後再將執行中的權重與 queue 第一個 process 的權重比較，如果 queue 裡的權重較高則將執行中的 process 踢出來換 queue 中第一個 process 執行

RR 的演算法：

按照 FIFO 的演算法，但每經過 500 的時間之後，如果當前的 process 並未執行完畢，就把當前的 process 踢出來，並丟到 queue 中的最後一個，並從 queue 中取第一個 process 出來執行。

2. 執行範例測資的結果

```
parallels@parallels-vm:~/Desktop/OSproject1$ sudo ./schedule
FIFO
3
P1 1 10
P2 2 5
P3 2 7
P1 6117
P2 6118
P3 6119
```

```
[  992.685697] [Project1] 6117 1556642505.852745665 1556642505.873800860
[  992.694590] [Project1] 6118 1556642505.874351978 1556642505.882730618
[  992.707056] [Project1] 6119 1556642505.883183404 1556642505.895204652
```