OS HW3 report

Name：荊姿芸
Student ID:0816091

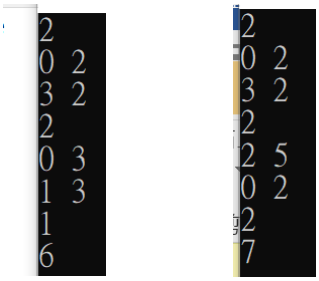
Question	Answer
Q1. Briefly describe about your data structure for recording process' time or anything you need to record.	Example of struct: `struct process { int id; int arrive_time; int burst_time; int waiting_time; int turnaround_time; };` Struct process is used to record info of process id, arrival_time, and burst_time from input. Arrival_time and burst_time are also stored in turnaround_time and waiting_time respectively in MLFQ and RR.(arrival->turnaround, burst->waiting) Burst_time is stored in remaining_time in SRTF. Processes are stored in vector<struct process> for further calculation.

Q2. How to simulate process scheduling?	SRTF: 1. Transfer process from input to srtf when it arrives. 2. Sort srtf by remaining_time of each process in it so that the first process in srtf(srtf[0]) has the shortest remaining_time. 3. Execute srtf[0] for time=1, remaining_time-=1,time+=1. 4. If remaining_time=0, the process is finish. Calculate its waiting_time=time-arrive_time-burst_time, turnaround_time=time-arrive_time. Transfer it from srtf to complete. 5. Repeatedly check for steps above until all processes are completed. RR: 1. Instead of transferring the process to another

	vector, input can be used directly. 2. Sort input by arrive_time so that input[0] has the smallest arrive_time; 3. After the process arrives, if its burst_time>q, it can only run for time=q. Then needs to pass to next process. Arrive_time+=q, time+=q, burst_time-=q. Push_back to the end of input then delete input[0] so that if a new process arrives at the time input[0] is terminated, the new process will have higher priority. Else if burst_time<=q, the process will finish at this execution. 4. Calculate its time+=burst_time, waiting_time=time-arrive_time-burst_time,

	turnaround_time= time-arrive_time. Transfer it from rr to complete. 5. Repeatedly check for steps 2~4 until all processes are completed. MLFQ: 1. Sort input by arrive_time so that input[0] has the smallest arrive_time; 2. Transfer process from input to higher priority queue(rr) when it arrives. 3. !input.empty(): there are processes not yet arrive. !rr.empty(): there are processes in higher priority queue that need to be execute first. if !input().empty&&!rr.empty() execute rr[0]. If its burst_time>q, it can only run for time=q. Then needs to pass to next

	process. Arrive_time+=q, time+=q, burst_time-=q. Transfer rr[0] to fcfs and sort fcfs by arrive_time. Else if burst_time<=q, the process will finish at this execution. Calculate its waiting_time=time-arrive_time-burst_time, turnaround_time=time-arrive_time. Transfer rr[0] to complete. Else if !input.empty()&&rr.empty()&&time<input[0].arrive_time: Process arrived has all run 1 time round robin and next new process has not yet arrive. Execute fcfs until it is preempted by rr(time=input[0].arrive_time) or burst_time==0. 4. If all process has

	arrived and run 1 time round robin, the rest of the processes are all in fcfs. 5. Repeatedly check for steps above until all processes are completed.
Q3. Some problems you meet and how to resolve.	1. In round robin where new process arrives at the same time when old process just leave. Wrong: Correct:  Solution: After running rr for time =t, push the process to the end of vector rr and delete rr[0]. In this case, the process just terminated will be at the end of the vector. And will be behind any process that has the same arrival

time after sorting by arrival time.

2. In SRTF and MLFQ, at first I didn't add the line highlighted. The process will terminate before finish running because input is empty and time>=input[0].arrive_time is not valid.
   Solution: Adding the highlighted line.

```
//if there are still process not yet arrived
if(!input.empty()){
    //add process from input to rr while it arrives
    if(time>=input[0].arrive_time){
        rr.push_back(input[0]);
        input.erase(input.begin());
    }
```

3. In MLFQ-FCFS, at first I think that fcfs might be preempted any time. So I change arrive_time and burst_time of fcfs after every execution and it causes some serious problems.
   Solution: By printing out every execution I found the problem and then change the

	method for fcfs. FCFS can only be executed in certain conditions: (1) every process has arrived and run rr 1 time. (2) old processes have all run rr 1 time and have leftovers. But new process has not arrived yet. Adding the conditions make fcfs works well.
Q4. What you learned from doing OS hw3 and something you want to discuss with TAs.	Writing a process scheduling program is very different than understand and write out how process scheduling works on paper. There are many details need to pay attention or else the program won't work as you wish it would. Is it possible to change the table format next time?左右兩格對用英文寫報告真的不太友善。會一直在奇怪的地方換行，閱讀起來也不是很方便。