Inner clock 전역 변수를 만들었다. 알고리즘들은 해당 전역변수에 synchronize된다.

- FIFO

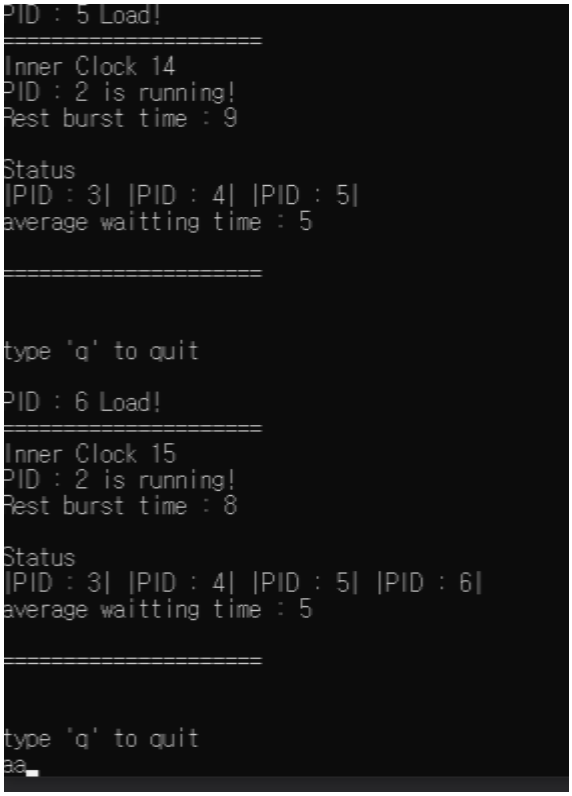
Process id를 Flag로 사용하고 있다.

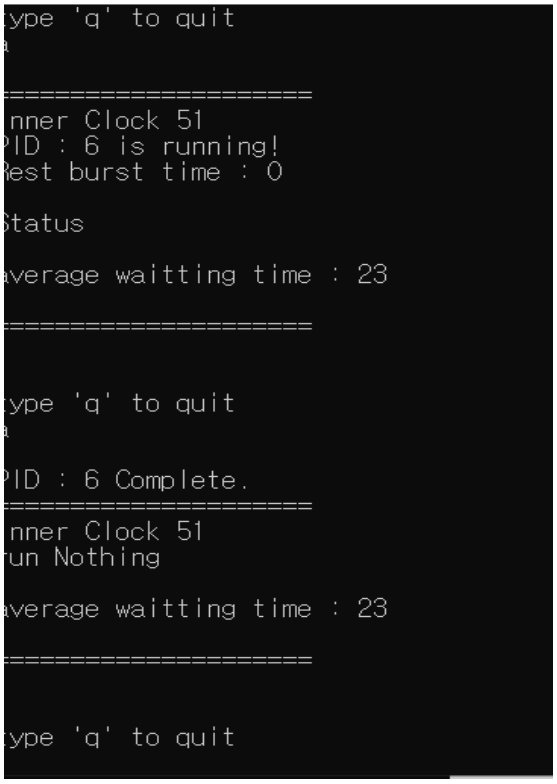
process_id가 -1 (존재하지 않음)면 queue에 load된 것 중 우선순위가 높은 맨 앞의 것을 고른다.

이후, process id가 등록하고, 해당 process가 끝날 때까지 실행한다.

Process가 끝날 때, process id를 -1로 바꿔 놓으면 다음 프로세스를 가져와서 실행한다.

결과:





Load할 시간이 같은 object들은 한번에 load된다.

- SRJF

Inner clock에 맞춰 rest time이 적은 원소를 priority queue에서 뺐다가 다시 넣는 형식으로 동작한다. C++ STL에 template specialize 했다.

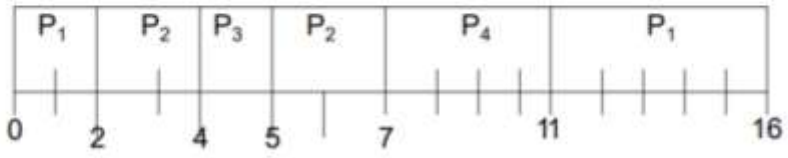
```
====================
Inner Clock 2
PID : 2 is running!
Rest burst time : 3

Status
|PID : 1|
average waitting time : 1

====================


type 'q' to quit
a

PID : 3 Load!
====================
Inner Clock 4
PID : 2 is running!
Rest burst time : 2

Status
|PID : 3| |PID : 2| |PID : 1|
average waitting time : 1

====================


type 'q' to quit
```

```
Rest burst time : 0

Status

average waitting time : 3

====================


type 'q' to quit
a

====================
Inner Clock 20
PID : 1 is running!
Rest burst time : 0

Status

average waitting time : 3

====================


type 'q' to quit
a
```

Example of Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

■ SJF (preemptive) (= SRTF)



■ Average waiting time = (9 + 1 + 0 +2)/4 = 3

해당 배열과 동일하게 나왔다.

- RR

Round Robin의 Sortest remaining job first와 다른 차이점은 해당 부분이다.

```
//SortTermScheduling
if(inner_clock % time_quantum)
    RoundRobin_algorithm.Dispatch();
inner_clock++;
}
```

Time Quantum마다 현재 실행중인 process를 Dispatch함수를 통해 다른 process로 바꾼다.

모든 process의 우선순위가 같다면 queue에 있는 모든 process를 수행해야 한다.

하지만 문제의 경우에는 Burst Time을 정확하게 예측할 수 있다.

그래서 queue안에 rest time으로 우선순위를 부여하여 처리하였다.

결과 :

```
ID : 5 Complete.
====================
nner Clock 59
ID : 5 is running!
est burst time : 0

tatus
PID : 2|
verage waitting time : 23

====================


ype 'q' to quit


====================
nner Clock 61
ID : 2 is running!
est burst time : 11

tatus

verage waitting time : 29

====================


ype 'q' to quit
```