

1. FIFO

Deque를 이용하여 처음 들어간 pcb가 끝날 때까지 context switching이 일어나지 않음.

```
PID : 5 Load!
=====
Inner Clock 15
PID : 2 is running!
Rest burst time : 0

Status
PID : 3| PID : 4| PID : 5| PID : 6|
average waiting time : 4
=====

type 'a' to quit
a

PID : 2 Complete:
=====
Inner Clock 23
Run Nothing
PID : 3| PID : 4| PID : 5| PID : 6|
average waiting time : 4
=====
```

2. RR

모든 process의 우선순위가 같다고 가정했다. Deque를 사용하여 구현했다.

매번, 3초 다 다른 process를 실행하기 위해,

```
int current_index = (inner_clock / time_quantum) % PCB_queue.size();
```

current_index 변수를 사용하여 접근하였다.

그리고 process의 rest burst time이 0이 되면, deque에서 해당 원소를 뺀다.

```
type 'a' to quit
a

PID : 5 Load!
=====
Inner Clock 15
PID : 5 is running!
Rest burst time : 7

Status
PID : 1| PID : 2| PID : 3| PID : 4| PID : 5| PID : 6|
average waiting time : 7
=====

type 'a' to quit
Context switching
=====
Inner Clock 18
PID : 6 is running!
Rest burst time : 7

Status
PID : 1| PID : 2| PID : 3| PID : 4| PID : 5| PID : 6|
average waiting time : 7
=====

type 'a' to quit
=====
Inner Clock 18
PID : 6 is running!
Rest burst time : 5

Status
PID : 1| PID : 2| PID : 3| PID : 4| PID : 5| PID : 6|
average waiting time : 8
=====

type 'a' to quit
a
```

3. SRJF

가장 이상적인 방법이다. Priority queue를 활용하여 구현했다. Process를 넣자마자 남은 시간이 적은 순서대로 순서가 정해진다.

```
Inner Clock 14
PID : 3 is running!
Rest burst time : 1

Status
[PID : 3] [PID : 4] [PID : 5] [PID : 2]
average waitting time : 2

=====

type 'q' to quit
q

PID : 6 Load!
=====
Inner Clock 15
PID : 3 is running!
Rest burst time : 0

Status
[PID : 3] [PID : 4] [PID : 6] [PID : 5] [PID : 2]
average waitting time : 2

=====

type 'q' to quit
q

PID : 3 Complete.
=====
Inner Clock 15
PID : 3 is running!
Rest burst time : 0

Status
[PID : 4] [PID : 6] [PID : 5] [PID : 2]
average waitting time : 2

=====

type 'q' to quit
q
```