Homework

Replacement

Let's consider about a page-removal algorithm: clock algorithm. Suppose we have a primary device which has 3 physical blocks, every time a reference string P come, it will follow the pseudo-code:

```
if hit(P)
  res_block = block contains P
  referenced_bit[res_block] ← True
  clock_arm does not change
else
  while referenced_bit[clock_arm]
    referenced_bit[clock_arm] ← False
    clock_arm ← next block
  res_block = block[clock_arm]
  referenced_bit[clock_arm] ← True
  clock_arm ← next block
return res_block
```

Fill in the table (If you don't know what to fill just write down a '-'). Note: '*' means the position of the clock arm; you also need to tell what the referenced bit is for each page block at that time (1 for True and 0 for False) .

Time	0	1	2	3	4	5	6	7	8
Reference string	-	3	4	2	6	4	3	7	4
Primary Device Contents	* - -	3 * -	3 4 *	3 * 4 2					
Referenced Bit	0 0 0	1 0 0	1 1 0	1 1 1					
Page Absent	-	Y	Y	Y					

Scheduling

We have following jobs in the workload. No I/O issues are involved.

Job	Arrival Time	Run time
A	0ms	4ms
B	1ms	1ms
C	4ms	5ms
D	6ms	2ms

- ✧ When a job arrives, it is added to the tail of the work queue.
- ✧ CPU picks job to run after all queue operations.
- ✧ The **MLFQ** policy has 2 priority queues, higher one with time-slice of 1ms and lower one with time-slice of 2ms. We use **RR** in each queue. Priority boost isn't supported.
- ✧ No preemption in **MLFQ**.
- ✧ We do RR by moving the **recently executed task** to the end of the queue.
- ✧ The priority of operations is RR movement > accepting new job.

Please calculate the **average** response time and **average** turnaround time for different scheduling policies.

Scheduling Policy	Turnaround Time	Response time
FIFO	[1]	[2]
STCF	[3]	[4]
MLFQ	[5]	[6]