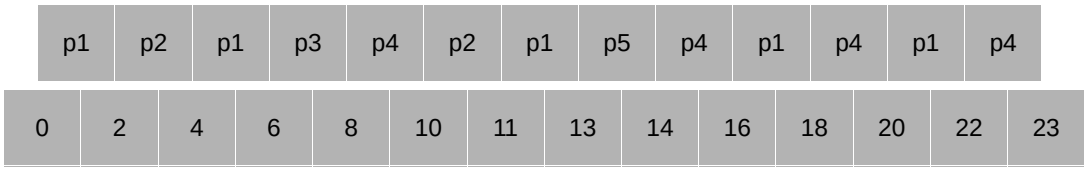
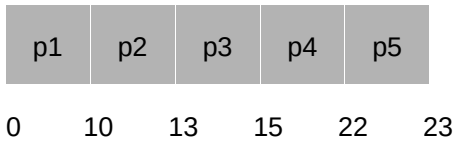
PART 1

ROUND ROBIN



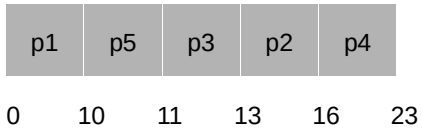
turn around time – 12.2+7cs
wait time – 7.6+7cs

FCFS



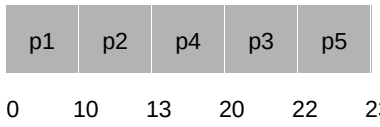
turn around time – 13.2+2cs
wait time – 8.6+2cs

SRTF



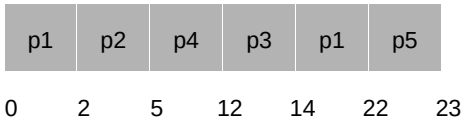
turn around time – 11.2+2cs
wait time – 6.6+2cs

priority



turn around time – 14.2+2cs
wait time – 9.6+2cs

priority with preemption



turn around time – 11.8+3cs
wait time – 7.2+3cs

2. If the block is in the cache the access time will be faster, if the cache is not stored on the disk. If the block is not in the cache (cache miss) it will take more time because we need to search the cache to determine if we hit or miss.

3. Caching is handled by the operating system. It saves data to the memory in contrast to page swapping that is handled by the memory management unit (MMU). The MMU has limited calculation power, it cannot store intermediate data and therefore it cannot preform complicated algorithms such as FBR.

4.
LRU better than LFU
cache size 2 blocks
file 3 blocks (A,B,C)
- read from A 10 times
- read from B 1 times
- read from C 1 times
- read from B 1 times
- read from C 1 times

LRU 3 miss, 11 hit
LFU 5 miss, 9 hit

LFU better than LRU
cache size 2 blocks
file 3 blocks (A,B,C)
- read from A 2 times
- read from B 1 times
- read from C 1 times
- read from A 1 times

LRU 4 miss, 1 hit
LFU 3 miss, 2 hit

a bad patten for both algorithms
cache size 2 blocks
file 3 blocks (A,B,C)
- read from A 1times
- read from B 1 times
- read from C 1 times
- read from A 1 times
- read from B 1 times
- read from C 1 times
- …

both algorithms might miss every access

5. in LFU a block can be stuck in the cache after have been accessed many time in a short time like in a loop. After the loop we stop accessing the block but because its high access count it will be priorities to stay in the cache for a long time, possibly forever. We solve this problem in the FBR by not increasing the block count if it is in the new section, it been recently accessed.

PART 2
1.
root directory guide → I-node of os → os directory guide → readme.txt I-node → first indirect → access data → update data → update readme.txt I-node.

8 disk access

2.
seek – system call → software interrupt
read – system call → software interrupt
reading from disk → hardware interrupt

3.
a. we will use FCFS with an exception.
When preemption occurs we will run the process for one quantum.

An a A type job ends after a quantum, because of the preemption it will have no waiting time in the algorithm. Only B type will have waiting time. This way we decrease the waiting time to minium.

b. no, we add extra CS over FCFS to decrease the waiting time. Classic FCFS will have less overhead