hassamoura rani.aboraya
hassan moura (205352115) rani aboraya(316396787)

EX: 3

Theoretical Part:

Part 1:

1.
a. RR:

p1	0-2
p2	2-4
p3	4-6
p4	6-8
p5	8-9
p1	9-11
p2	11-12
p4	12-14
p1	14-16
p4	16-18
p1	18-20
p4	20-21
p1	21-23

Turnaround time ==> 12.6
average wait time ==> 7.75

b. FCFS:

p1	0-10
p2	10-13
p3	13-15
p4	15-22
p5	22-23

Turnaround time ==> 16.6
average wait time ==> 8

c. SRTF:

P1	0-10
P5	10-11
P3	11-13
P2	13-16
P4	16-23

Turnaround time ==>14.6
average wait time ==> 8

d. PRIORITY SCHEDULING

P1	0-10
P5	10-11
P3	11-13
P2	13-16
P4	16-23

Turnaround time ==>14.6
average wait time ==>8

e. PRIORITY SCHEDULING WITH PREEMPTION

P1	0-7
P5	7-8
P1	8-11
P3	11-14
P2	14-16
p4	16-23

Turnaround time ==> 13.1
average wait time ==> 8

2.
in case we want to disposable read file,
so we want to access the data once for read and then never
use it again, so instead of accessing the disk one time and read the file
from it, in case we have a cache with small memory we need to access the disk
many times to put the blocks in cache memory and we need to delete blocks
from cache if it become full so all this process is not do it faster
reading of the file.

3.

we cant hold the LRU of each blocks,in order to the hardware that cant supporting the time of usage of the cached algorithm except to the single used bit that used of the cached algorithm.

4.

in the one hand LRU ,using a block a lot of time , its counter will be added along of used, and according to count of uses it will stay in the cache until it becomes the least count in the cache so will be retrieved ,in the other hand we have the LFU that will remove it according the least frequently used and this the feature of the LFU so in this working pattern the LRU is better than the LFU. other working Pattern that show the opposite, let us have a 9 blocks to manipulate and the cache size is 5, if we iterating by the algorithm of LRU we will pull a new block from the disk, but LFU has saved at least one .finally with working pattern that don't help all, in case the cache can keep two files and in the same time read three files over and over again so in LFU when we try to access some files, that file was already removed because it was the least frequently used just before, so we will have cache misses once again and in the LRU the last file will always be removed right before we read it again therefore we will have cache miss.

5.
if we increased the accesses to blocks in the new section, we will have a problem, certain blocks are relatively infrequently referenced overall, and yet when they are referenced, due to locality there are short intervals of repeated re-references, thus building up high reference counts. After such an interval is over, the high reference count is misleading: it is due to locality, and cannot be used to estimate the probability that such a block will be re-referenced following the end of this interval

Part 2:

1.
a. access to inode of "/".
b. access to inode of "os".
c. access to inode of "README".
d. access to inode of the file
e. access to the pointer clock of the single direct

2.

the "seek" command its system call which leads to software interrupt.
the "read" command its system call which leads to software interrupt.

3.

a. using scheduler of multi level feedback with two queue A and B/
queue A ==> run by Roun
queue B ==> run by SRTF
all the jobs from the A queue will insert to B queue.
we will give the primary with preemption to the jobs from the queue.

b. no, in order to using the preemption