# 2017 Operating System Examination Solution

1. a. i) Shortest Remaining Time First. Notice that Shortest Job First is incorrect here because it's not preemptive.

   ii) FCFS:

   System throughput : $\frac{3}{100+1+10} = \frac{1}{37}$

   Avg. turnaround time : $\frac{100+101+111}{3} = 104$

   SJF:

   System throughput : $\frac{3}{1+10+100} = \frac{1}{37}$

   Avg. turnaround time : $\frac{1+11+111}{3} = 41$

   iii) None of them are predictable. In FCFS, the turnaround time of a job depends on how many jobs are ahead of it. In RR, the turnaround time actually depends on how many jobs are there in the system. In priority scheduling, the turnaround time of a job depends on its priority: if it has a low priority, then it might be delayed indefinitely.

   iv) No. Two virtual machines have their own schedulers and we cannot schedule across different operating systems. Notice that it is possible for the host to set the priority of different virtual machines, not individual processes.

   b. i) Below is a deadlock situation

```
// code for P1
int main() {
  lock(R1);
  lock(R2);
  // use R1 and R2
  unlock(R2);
  unlock(R1);
}
```

```
// code for P2
int main() {
  lock(R2);
  lock(R1);
  // use R1 and R2
  unlock(R1);
  unlock(R2);
}
```

   When P1 executes, it first acquires R1's lock; when P2 executes, it first acquires R2's lock. When they execute concurrently, it will result in P1 waits for P2 and P2 waits for P1.

ii) Below is a code sequence that never triggers a deadlock

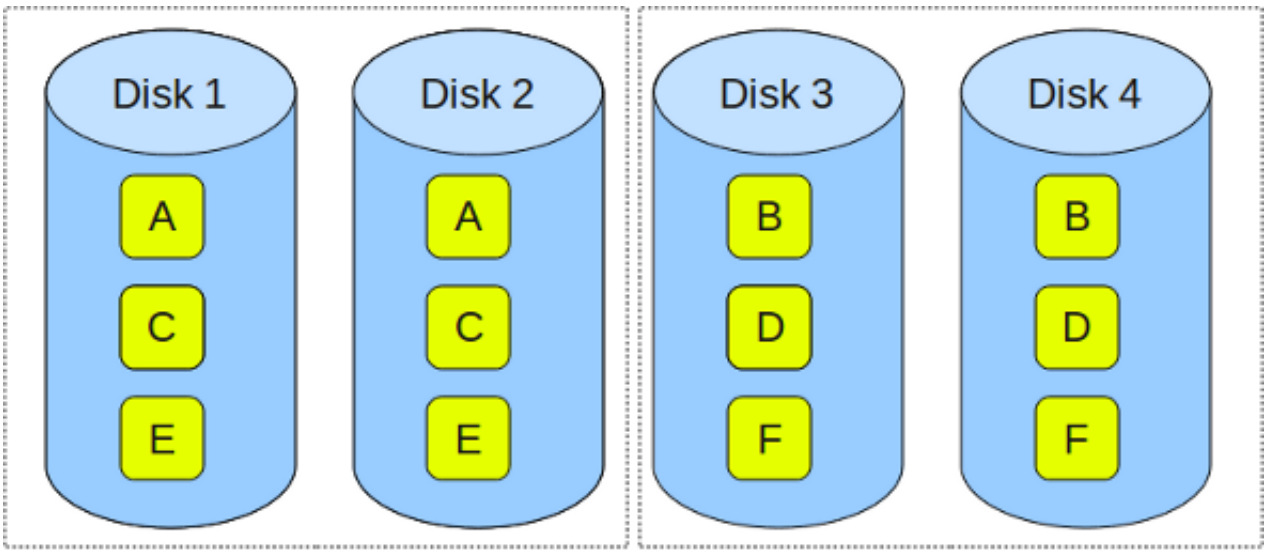
```
// code for P1
int main() {
  lock(R1);
  lock(R2);
  // use R1 and R2
  unlock(R2);
  unlock(R1);
}
```

```
// code for P2
int main() {
  lock(R1);
  lock(R2);
  // use R1 and R2
  unlock(R2);
  unlock(R1);
}
```

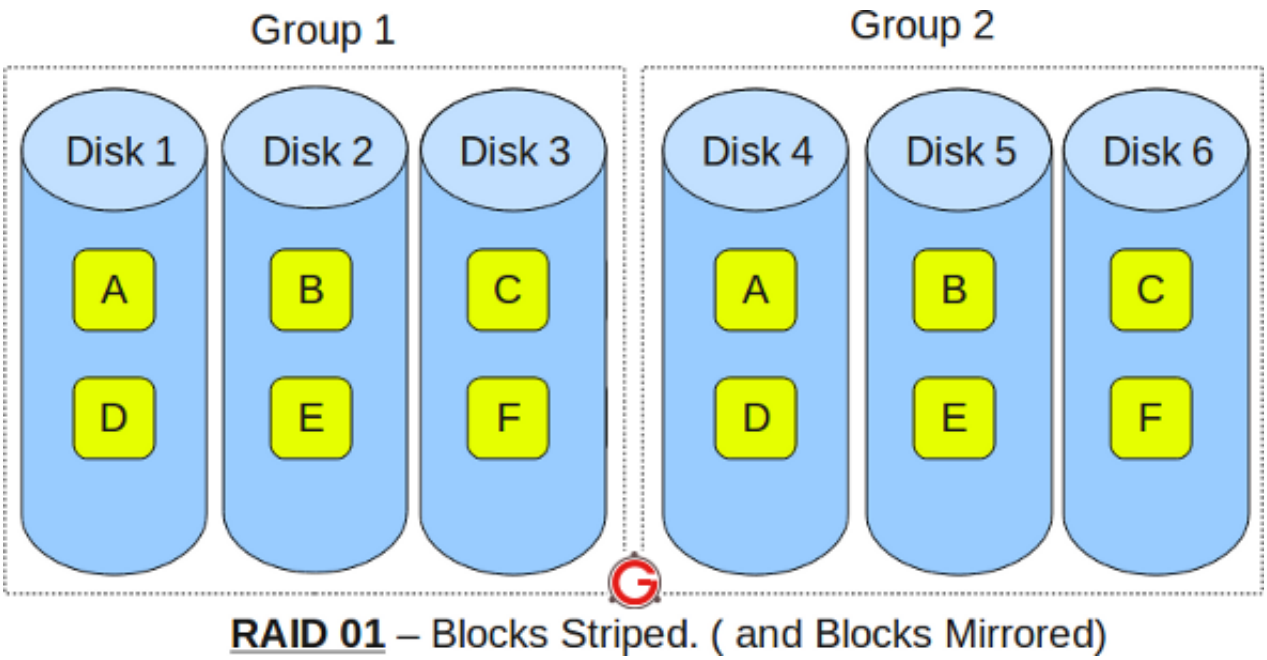
c. i) RAID-0 will give the highest throughput because RAID-1 needs to write data to mirror disks while RAID-0 does not need to perform such operation.

ii) RAID-0 : $GN$, RAID-1 : $\frac{GN}{2}$

iii) We can use a hybrid approach of RAID-0 and RAID-1. We can have two disks grouped as a single RAID-1 group and among those RAID-1 groups, the disks behave like RAID-0. Alternatively we can group them as RAID-0 groups and use RAID-1 outside. Below are two diagrams illustrate these two ideas.




**RAID 10** – Blocks Mirrored. ( and Blocks Striped)




2. a. The first column is the virtual addresses in the page table; the second column is the dirty bit representing whether the page has been written by any process; the third column is valid bit indicating whether the page is in the memory(1), or is in the swap disk or invalid(0).

   b. The page size is $2^{10} = 1024 = 1$ KBytes.

   c. The maximum amount of physical memory supported is $1 \cdot 2^{16-10} = 2^6 = 64$ KBytes.

   d. Before translation, we first expand the entries in the page table into binary

$$\texttt{0x1800} = 000110\ 0000000000$$
$$\texttt{0x4C00} = 010011\ 0000000000$$
$$\texttt{0x5400} = 010101\ 0000000000$$
$$\texttt{0x7C00} = 011111\ 0000000000$$
$$\texttt{0x1400} = 000101\ 0000000000$$

   i) $\texttt{0xF1F} = 000011\ 1100011111 \Rightarrow 011111\ 1100011111 = \texttt{0x7F1F}$

   ii) $\texttt{0xC000} = 110000\ 0000000000 \Rightarrow$ No entry at index $110000 = 48$, cannot translate

   iii) $\texttt{0x403} = 000001\ 0000000011 \Rightarrow 010011\ 0000000011 = \texttt{0x4C03}$

   iv) $\texttt{0x0} = 000000\ 0000000000 \Rightarrow 000110\ 0000000000 = \texttt{0x1800}$

   e. A more compacted version of the page table would incorporate the dirty and valid bit into the last ten digits, with the least significant bit represents valid-invalid bit and the second least significant bit represents the dirty bit.

$$\texttt{0x1803} \ \ (\texttt{0x1800} + \texttt{0000000011})$$
$$\texttt{0x4C03} \ \ (\texttt{0x4C00} + \texttt{0000000011})$$
$$\texttt{0x5400} \ \ (\texttt{0x1800} + \texttt{0000000000})$$
$$\texttt{0x7C03} \ \ (\texttt{0x1800} + \texttt{0000000011})$$
$$\texttt{0x1402} \ \ (\texttt{0x1800} + \texttt{0000000010})$$

The new page table will only have a 16-bit physical address as its entry with the least significant bit represents valid-invalid bit and the second least significant bit represents the dirty bit.

f.