CS314 Endsem Answer Key:

Q1. **All questions except G are of 1mark. G is of 2marks**

A. Timer interrupts a useful mechanism for the OS ? Why?

Allows OS to keep control of the CPU. Also allows implementation of certain scheduling policies.

B. Operating systems frequently exploit locality to improve performance. Briefly describe two examples where operating systems do so, and state how locality is exploited.

Virtual memory and associated page replacement algorithms exploit the temporal and spatial lo-cality of a program's accesses to its data. Programs tend to access pages that were accessed recently,enabling physical memory to be used as a cache for all of the data used by a program during its execution. Programs also tend to access data across a given page,which helps amortize the cost of bringing pages in from disk. Without such locality, virtual memory would be prohibitively slow
2. The file buffer cache and associate replacement algorithms exploit the temporal and spatial locality of a program's accesses to file blocks. Programs tend to access recently-accessed file blocks, enabling the file buffer cache to serve a significant fraction of requests from memory instead of disk even though the file system is a couple of orders of magnitude larger than the file buffer cache. Programs also tend to access file blocks logically near previously-accessed file blocks (e.g., sequentially), enabling the system to use techniques like read-ahead to anticipate program behavior.And there are others (e.g., TLB)...

C. Why would a scheduling algorithm that gave priority to processes based on how much time they have consumed (i.e. processes whose total CPU time is higher have higher priority) be a bad idea?

Using CPU time gives your more time. This promotes a counter strategy of starting a process and running for a long time so that it will have a high priority when it is ready to do real work. Also, leads to starvation of other processes.

D. Assume there exists three-handed Philosophers. Consider a table with five three-handed philosophers and a pile of N chopsticks in the middle of the table. Each philosopher needs 3 chopsticks to eat. What is the smallest N such that deadlock is impossible?

With 10 chopsticks, each philosopher could pick up two chopsticks and end up deadlocked. With 11, there is always at least one philosopher who can eat.

E. Explain the four necessary conditions for deadlock.
Circular Wait, Hold & Wait, No Preemption, Mutual Exclusion.

F. Explain how the Resource-Request (Banker's) algorithm prevents deadlock.

It ensures that there is always a way to meet the resource requirements of the processes in the system (i.e. there exists a safe sequence) without getting into deadlock. When a resource request is given, the system pretends that the request has been satisfied, and verifies that there still exists a safe sequence that allows all existing processes to make their remaining resource requests.

G. Explain what the clock algorithm for page aging is and explain why it is often used instead of full LRU.

Periodically a sweep is made of all the page frames with access bit set and they are marked with the current value of a counter (called the clock). The clock is then advanced and all access bits are cleared. This is used to group pages into equivalence classes of ages since last access. Having the hardware set a use bit and the OS make the sweep requires much simpler hardware than maintaining full hardware LRU.

LRU is considered to be a reasonably good algorithm • Problem is in implementing it efficiently • Hardware implementation: counter per page, copied per memory reference, have to search pages on page replacement to find oldest, Every time page is referenced, save system clock into the counter of the page• Software implementation: no search, but pointer swap on each memory reference, high contention • In practice, settle for efficient approximate LRU • Find an old page, but not necessarily the oldest • LRU is approximation anyway, so approximate more

H. What are the different ways an OS can decide when a page (or pages) on disk should be brought into memory?
Demand Paging, Anticipatory paging, Hybrid

I. With a 15000 RPM disk, what is the expected average rotation time for a random access
Ans 2 ms - 4ms is the full rotation time; the expected rotation time for a random access will be 1⁄2 of this full amount. 60 sec / 1 min * 1 min / 15000 revs * 1000 ms / 1 sec = 4 ms per full rotation.

J. Is SPTF (Shortest positioning time first) scheduling easier to implement inside of a disk than within the OS. Explain your answer.
OS doesnt know geometry so SPTF is easier to implement inside disk

K. Why does the peripheral bus used by hard disk drives provide lower bandwidth than the memory bus?
Device Cost and Physics - busses that are further away from the main CPU tend to have lower bandwidth and the devices that are connected to them tend to be slower (than RAM).

2. The following set of questions ask you to translate logical read and write operations performed on top of a RAID system to the physical read and write operations that will be required to the underlying disks. Specifically, for each RAID configuration, translate the logical requests to the physical operations performed on the correct disk number and physical block address (offset). In all cases assume a block and chunk size of 4 KB.
[Answers are expected in the following format : Read from disk 7, offset 7 ; Write to disk 1 and disk 5 at offset 7]

1. RAID Level: 0; Number of Disks: 8; Random Read from logical block number 58
2. RAID Level: 1; Number of Disks: 8; Random Write to logical block 29
3. RAID Level: 4; Number of Disks: 4; Random Write to logical block 50
4. RAID Level: 5; Number of Disks: 4; Random Read from logical block 10

1. RAID-0 is simple striping;

58 % 8 (number of disks) = 2 -> disk 2
58 / 8 (integer division) = 7 -> block offset 7
Read from disk 2, offset 7

2. RAID-1 is mirroring; with 8 disks, it is like having 4 mirrored pairs

29 % 4 = 1 -> mirrored pair number 1, which are disks 2, 3
29 / 4 = 7 -> offset 7
Write to disk 2 and disk 3 at offset 7

3. Since this is a random write, the best approach is to read the old data and the old parity and then flip the parity for every bit that is changed in the new data.

With RAID-4, disks 0, 1, 2 are used for data; disk 3 for parity.
50 / 3 (number of data disks) = 16 -> offset 16
50 % 3 = 2 -> disk 2 for data
Read from disk 2 and 3, offset 16; Write to disk 2 and 3, offset 16.

4. **Read from disk 2, offset 3**

3. Consider a disk with the following characteristics:
• Number of surfaces: 8 (= 2^3)
• Number of tracks / surface: 512 K (= 2^19)
• Number of bytes / track: 8 MB (= 2^23 bytes)
• Number of sectors / track: 8 K (= 2^13)

• On-disk cache: 16 MB (= 2^24 bytes)

1. How many heads does this disk have? [**8 heads: 1 head per surface]**
2. What is the size of each sector? **[2^23 bytes / track * track/2^13 sectors = 2^10 bytes /sector]**
3. How many bytes per cylinder? **[2^23 bytes / track * 8 tracks / cylinder = 2^26 bytes / cylinder]**
4. What is the total capacity of this disk? [**2^26 bytes / cylinder * 2^19 cylinders = 2^45 bytes]**

4. Assume a situation in which each process has three frames. Suppose the page reference string of some process is 0, 7, 2, 0, 7, 1, 0, 3, 1, 2Initially no pages are mapped to physical frames. Now consider the state of the process's page table after the first 7 references (i.e., after page accesses 0 7 2 0 7 1 0). Which (up to three) pages are mapped at this time assuming one of the following page replacement schemes, and how many page faults have occurred then. Also show in the last column how many page faults occur in total after all 10 references?

Scheme	all page numbers of mapped pages **after 7 references** (3 max.)	#page faults **after 7 references**	#page faults total **(after 10 references)**
First In First Out (by way of example)	0 1 2	5	7
LRU (Least Recently Used)	0 1 7	4	6
OPT (Belady)	0 1 2	4	5

5.

```
int a = 0;
int main() {
    fork();
    a++;
    fork();
    a++;
    if (fork() == 0) {
        printf("Hello!\n");
    } else {
         printf("Goodbye!\n");
    }
    a++;
    printf("a is %d\n", a);
 }
```

How many times will the message “Hello!\n” be displayed?
4 -- The first call to fork() results in two processes; each of those two processes then calls fork() which results in a total of four processes; the next call to fork() results in eight processes, but only the child process (of which there are four) prints “Hello”.

What will be the final value of “a” as displayed by the final line of the program?
Each process has its own copy of the variable a, with no sharing across processes. Each process will increment ‘a’three times, resulting in an identical value of **‘a=3’** for all 8 processes

6. Assume three jobs arrive at approximately the same time, but Job A arrives slightly before Job B, and Job B arrives slightly before job C. Job A requires 2 sec of CPU, Job B is 8 secs, and Job C is 7 secs. Assume a time-slice of 1 sec.

a) Given a FIFO scheduler, what is the turnaround time of job B?
b) Given a FIFO scheduler, what is the average response time of the three jobs?
c) Given a RR scheduler, what is the turnaround time of job B?
d) Given a RR scheduler, what is the average response time of the three jobs?
e) Given a SJF scheduler, what is the turnaround time of job B?
f) Given a SJF scheduler, what is the average response time of the three jobs?

Answer :
a) FIFO: A (until 2), B (until 10), C (until 17); B completes at time 10 seconds
b) Average response time: (0 + 2 + 10) / 3 = 4 seconds
c) B has the longest CPU burst and with RR will finish after every other job finishes, which is 2 + 8 + 7 = 17 seconds. RR schedule: ABCABCBCBCBCBCBCB
d) Average response time: (0 + 1 + 2) / 3 = 1 second.
e) B is longest, so it is scheduled last; finishes when workload ends: 2 + 7 + 8 = 17.
f) Average response time: (0 + 2 + 9) / 3 = 3.67

Q7.

Consider a memory allocator that uses the buddy allocation algorithm to satisfy memory requests. The allocator starts with a heap of size 4KB (4096 bytes). The following requests are made to the allocator by the user program (all sizes requested are in bytes): ptr1 = malloc(500); ptr2 = malloc(200); ptr3 = malloc(800); ptr4 = malloc(1500). Assume that the header added by the allocator is less than 10 bytes in size. You can make any assumption about the implementation of the buddy allocation algorithm that is consistent with the description in class.

(a) Draw a figure showing the status of the heap after these 4 allocations complete. Your figure must show which portions of the heap are assigned and which are free, including the sizes of the various allocated and free blocks.

(b) Now, suppose the user program frees up memory allocations of ptr2, ptr3, and ptr4. Draw a figure showing the status of the heap once again, after the memory is freed up and the allocation algorithm has had a chance to do any possible coalescing.

Ans:

(a) [512 B][256 B] 256 B free [1024 B][2048 B]

(b) [512 B] 512 B free, 1024 B free, 2048 B free. No further coalescing is possible.

8.

Consider a system where each process has a virtual address space of 2^v bytes. The physical address space of the system is 2^p bytes, and the page size is 2^k bytes. The size of each page table entry is 2^e bytes. The system uses hierarchical paging with l levels of page tables, where the page table entries in the last level point to the actual physical pages of the process. Assume $l \geq 2$. Let v_0 denote the number of (most significant) bits of the virtual address that are used as an index into the outermost page table during address translation.

(a) What is the number of logical pages of a process?

(b) What is the number of physical frames in the system?

(c) What is the number of PTEs that can be stored in a page?

(d) How many pages are required to store the innermost PTEs?

Ans:

(a) 2^{v-k}

(b) 2^{p-k}

(c) 2^{k-e}

(d) $2^{v-k} / 2^{k-e} = 2^{v+e-2k}$

9.

Imagine the following commands are run on a file system that supports hard and soft links.

```
echo "file" > file
ln file file2
echo "file2" >> file2
mv file file3
echo "file3" >> file3
```

a) What output will you see if you run cat file2(assume nothing wrong with whitespace or carriage returns)?

[**file\n file2\n file3\n** ; Imagine "file" points to a newly allocated inode, number I. Since a hard link is used, "file2" will refer to this same inode. Changing the name "file" to "file3" in the directory does nothing to the actual contents of inode I. It does not matter which name is used to refer to inode I.]

b) What output will you see if you run cat file3 (assume nothing wrong with whitespace or carriage returns)? **[file\n file2\n file3\n]**

Next, the following commands are run

```
echo "bar" > bar
ln -s bar bar2
echo "bar2" >> bar2
mv bar bar3
echo "bar3" >> bar3
```

a) What output will you see (assume nothing wrong with whitespace or carriage returns) if you run cat bar2 ? [**bar2: No such file or directory,** Since bar2 is only a soft (or symbolic) link, bar2 actually points to the pathname "bar" (and not directly to the same inode). As a result, if bar is deleted or renamed, bar2 will not point to anything valid.]

b) What output will you see (assume nothing wrong with whitespace or carriage returns) if you run cat bar3 ? [**bar\n bar2\n bar3\n;** When the command "echo bar2 >> bar2" was executed, bar2 pointed correctly to bar.]

Q.10

A. Two processes reading from the same virtual address will access the same contents.

False: processes have different address spaces, so different contents

B. The convoy effect occurs when longer jobs must wait for shorter jobs.

False: occurs when shorter jobs must wait for longer jobs

C. On a uniprocessor system, there may only be one ready process at any point in time.

False; just one RUNNING process, but many could be ready and want to be scheduled.

D. With producer/consumer relationships and a finite-sized circular shared buffer, consuming threads must wait until there is an empty element of the buffer.

False; consumers must wait until there is a full element.

E. A disadvantage of segmentation is that different portions of an address space cannot grow independently - [True/False]

False; segmentation lets each segment grow independently

F. The size of a virtual page is always identical to the size of a physical page [True/False]

True

G. FIFO with N+1 pages of memory always performs as well or better than FIFO with N pages of memory.

[False, see Belady's anomaly]

H.When interacting with a fast device, it can be better to spin wait than to use interrupts.

True; if the device responds very quickly, the response might come back faster than the time required to context-switch to another process and back again.

I. A disadvantage of the SCAN and C-SCAN scheduling algorithms are that they ignore the influence of rotation time on positioning cost.

[True; SCAN and C-SCAN just schedule based on the track (or cylinder) number.]

J. RAID-4 has better capacity than RAID-1 and better reliability than RAID-0.

True; RAID-4 has just one parity disk for N data disks, whereas RAID-1 has an extra mirror for every data disk; RAID-0 cannot withstand any disk failures, whereas RAID-4 can handle any single disk failing.

11.

Ans: (a) New inode allocated on disk (with link count=1), and inode bitmap updated in the process. (b) Directory entry added to parent directory, to add mapping from file name to inode number. (c) In-memory inode allocated. (d) System-wide open file table points to in-memory inode. (e) Per-process file descriptor table points to open file table entry.

12. Given a basic spin lock, assume that locking the spin lock takes A time units (if no one is holding the lock); unlock also takes A time units. Assume that a context switch takes C time units and a time slice is T time units long.

Assume the following code is executed by two threads on one processor (uniprocessor) roughly at the same time.

```
mutex_lock();
```

```
do_something(); // takes no time to execute
mutex_unlock();
exit();                 // takes no time to execute
```

a) What is the best-case time for the two threads on one CPU to finish executing this code sequence? [4A+C]