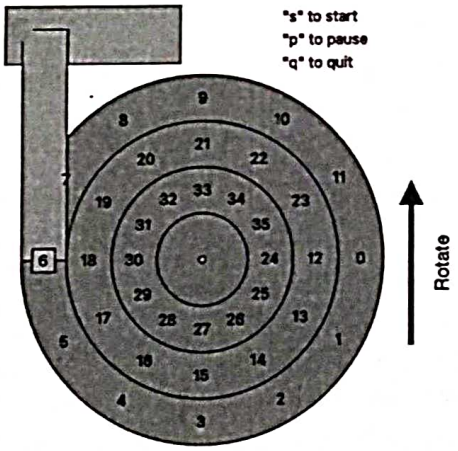
CS 304 Operating Systems End-Semester Exam			Marks obtained ↓
Date: 21.04.2023,	Total questions: 5	Total Marks: 40	
Name:	Roll No:		Time: 180 min

Question:	1	2	3	4	5	Total
Points:	10	5	11	3	11	40
Score:						

Instructions:

1. **You are not allowed to carry any electronic gadgets including calculators and mobile phones.**
2. **Please note, merely writing the answer without justification/ reason (wherever explicitly asked to give reason) will not fetch FULL marks.**
3. **Answer to all the parts of a question needs to be written together and the final answer needs to be written with a pen.**

10 1. (A) Given the disk in the figure below (which rotates counter-clockwise) and assuming the requests to the disk are as follows **13, 4, 35, 34, 5**. Answer the following questions:



(i) Assuming **FCFS** (first-come first-served) disk scheduling, which request would be serviced last?

(ii) Assuming a **SSTF** (shortest seek time first) disk scheduler, and a VERY FAST (essentially infinitely fast) seek; which request would be serviced last? (assume that when you get to a particular track, the scheduler will read the blocks in the fastest order available on that track)

(iii) Assuming a **SATF** (shortest access time first) scheduler, and a VERY FAST (essentially infinitely fast) seek; which request will be serviced last?

(iv) Assuming a VERY FAST seek time (essentially infinitely fast), how would you order the performance of disk schedulers (FCFS, SSTF and SATF) for those five requests, from fastest to slowest?

(B) Answer the following questions regarding RAID based systems:

(i) Generally, a mirrored RAID system (with mirroring level of 2) can tolerate 1 disk failure for certain, and up to _____failures depending on which disks fail." [Assume here there are N disks in the system]

(ii) The maximum bandwidth obtained during sequential writing to a 2-way mirrored array is _____[Assume here there are N disks, and that a single disk delivers S MB/s of disk bandwidth]

(iii) RAID-4 uses a disk for parity information for every group of disks it is protecting. Thus, th useful capacity is _____[Assume N disks, and B bytes of data per disk]

(iv) A single write (to a RAID-4) requires _____read(s) and then _____write(s) to the underlying disks. **[assuming subtractive parity]**

(v) Assuming that each disk, under a random write workload, delivers R MB/s, a RAID-5 system with N disks will deliver _____MB/s under a random write workload.

(vi) RAID Level(s) _____encounter(s) the 'small write' problem.

(C) For the questions given below, you need to compute how long it takes to perform **12 writes to random locations** within a RAID. Assume that these random writes are spread evenly across the disks of the RAID. Assume that each read or write takes **"D" time units.**

(i) How long does it take on a system consisting of **4 disks with RAID-0** ?

(ii) How long does it take on a system consisting of **4 disks with RAID-1** ?

(iii) How long does it take on a system consisting of **4 disks with RAID-4** ?

(iv) How long does it take on a system consisting of **4 disks with RAID-5** ?

(D) Answer the following questions to the point :

(i) Provide one reason why a DMA-enabled device driver usually gives better performance over a non-DMA interrupt-driven device driver.

(ii) What is a block device? What is a character device? Give examples where one interface be more appropriate than the other?

(iii) What are the techniques that can be used to guard against buffer overflow attacks?

(iv) What are access control lists ? How is it implemented in UNIX ?

(v) With a 15000 RPM disk, what is the expected average rotation time for a random access ?

(vi) Rather than writing updated files to disk immediately when they are closed, many UNIX systems use a delayed write-behind policy in which dirty disk blocks are flushed to disk once every X seconds. List one advantage and one disadvantage of such a scheme.

5 2. Suppose you have a disk partition of size 2^{40} bytes. It is subdivided into blocks of 8192 bytes. Ansuwer the following:

(A) How many blocks are there on the disk partition?

(B) You have a unix-like file system on the partition, with one superblock in position 0, followed by a sequence of blocks filled with i-nodes. Each i-node is 128 bytes. You want to have enough i-nodes to store 2^{20} files. How many i-node blocks do you need to store all these i-nodes?

(C) A block pointer identifies a block on the partition, and is 4 bytes long. Assume an i-node contains 13 block pointers. The first 10 point to the first 10 data blocks. The next three point to an indirect block, a double indirect block, and a triple indirect block. Also, the maximum file size can be approximated by just the number of data blocks reachable from the triple indirect block pointer (the rest is negligible). How much data (in bytes) could be accessed from the triple indirect block pointer in the i-node? **For this part of the question, assume the size of the block as 8192 bytes and the size of the disk is unbounded.**

(D) Assume now that the file system cache contains only the superblock. Assume the file with i-node #2015 has the string "Hello World" in it (that is, the file is just 11 bytes long). How many disk accesses would be necessary to read the contents of this file, given that you (and the kernel) know the i-node number? Explain

(E) Suppose the file's i-node number has to be retrieved first. Assume the name of the file is **/etc/test.txt**. Assume that the contents of each directory fits in a single block. The root directory / is described in i-node #2. Assume /etc is in i-node #5 (you know this, but the kernel doesn't know). Again, assuming only the superblock is in the cache and a cache large enough so the same block never has to be read more than once, how many disk accesses are required to read the file? Explain

(F) File **/etc/OS.txt** (which you know to be in i-node #7, but the kernel doesn't know) contains the details of operating systems concepts (size of the file is 4 MB). Assuming only the superblock is in the cache, how many disk accesses are required to retrieve the whole data? Explain.

(G) Suppose you want to add the text "A very good book." to the end of the **/etc/OS.txt** (the new text will be contained in a new data block at the very end). Suppose that the file system has only the superblock in its cache and can allocate free blocks without going to the disk. How many disk reads and how many disk writes are necessary (assuming a "write-through" cache? (Assume that among the i-nodes only i-node #7 has to be updated for this operation.) Explain.

11 3. (A) Consider the following segment table:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical addresses for each of the following logical addresses **[Segment, Offset]**? All values are in decimal.

(i) 1, 20
(ii) 2, 80
(iii) 3, 450
(iv) 6, 90

(B) Assume a page reference string for a process with m frames (initially all empty). The page reference string has length p with n distinct page numbers occurring in it. For any page-replacement algorithms, (a) What is the lower bound on the number of page faults? (b) What is the upper bound on the number of page faults? Justify your answer

(C) Consider a memory system with a cache access time of 10ns and a memory access time of 200ns. If the effective access time is 10% greater than the cache access time, what is the cache hit ratio H? (Fractional answers are okay).

(D) Suppose you have a system with virtual and physical memory address space of 32 bits and 4KB page size. Suppose you know that there will only be 4 processes running at the same time, each with a Resident Set Size of 512MB and a Working Set Size of 256KB. What is the minimum amount of TLB entries that your system would need to support to be able to map/cache the working set size for one process? What happens if you have more entries? What happens if you have less entries than the minimum?

(E) Assuming a page size of 1 KB and that each page table entry (PTE) takes 4 bytes, how many levels of page tables would be required to map a 34-bit address if every page table fits into a single page. Be explicit in your explanation.

(F) A process has **four page** frames allocated to it. The table below includes the following items: (1) the time of the last loading of a page into each frame, (2) the time of last access to a page in each frame, (3) the virtual page number present in each frame, and (4) the reference (R) and modified (M) bits for each frame. The times are in clock ticks, from the process starts at time zero to the event.

Virtual page No.	Page Frame	Time Loaded	Time Accessed(used)	R-bit	M-bit
2	0	60	150	0	1
1	1	130	160	0	0
0	2	70	155	1	0
3	3	100	145	1	1

A page fault to virtual page 4 has occurred. Which page frame will have its content replaced for each of the following page replacement algorithms: FIFO, LRU and Clock? Explain why in each case. Also, for the clock algorithm, explain for all the scenarios, the clock pointer can point to (page frame 0/1/2/3)

3 4. Consider the following snapshot of a system in which five resources A, B, C, D and E are available. Th system contains 2 instances of A, 1 of resource B, 1 of resource C, 2 resource D and 1 of resource E **after current allocation.**

	Allocation					*Request*					*Available*				
	A	*B*	*C*	*D*	*E*	*A*	*B*	*C*	*D*	*E*	*A*	*B*	*C*	*D*	*E*
P_0	1	0	1	1	0	0	1	0	0	1	2	1	1	2	1
P_1	1	1	0	0	0	0	0	1	0	1					
P_2	0	0	0	1	0	0	0	0	0	1					
P_3	0	0	0	0	0	1	0	1	0	1					

(i) Convert this matrix representation to a resource allocation graph.

(ii) Use the deadlock detection algorithm to determine whether the system contains a deadlock. If it is deadlocked, which processes are involved in the deadlock? Provide your detailed calculation

(iii) What is the difference between deadlock prevention and deadlock avoidance? What category does Bankers algorithm falls in and why?

11 5. (A) Assume we have three jobs that enter a system and need to be scheduled. The first job that enters is called A, and it needs 10 seconds of CPU time. The second, which arrives just after A, is called B, and it needs 15 seconds of CPU time. The third, C, arrives just after B, and needs 10 seconds of CPU time. `For all questions below that are involving round-robin, assume that there is no cost to context switching.` Also assume that if job X arrives just before Y, a round-robin scheduler will schedule X before Y.

(i) Assuming a shortest-job-first (SJF) policy, at what time does B finish?

(ii) Assuming a longest-job-first (LJF) policy, at what time does B finish?

(iii) Assuming a round-robin policy (with a time slice length of 1 second), when does job A finish?

(iv) Assuming a round-robin policy (with a time-slice length of 1 second), when does job B finish?

(v) Assuming a round-robin policy (with an unknown time-slice which is some value less than or equal to 2 seconds), when does job B finish?

(vi) Assuming a round-robin policy **(with an unknown time-slice)**, for what values of the time-slice will B finish before C?

(B) What will the values of pid at lines A, B, C and D. Assume that the **actual pids of the parent and child are 2600 and 2603**, respectively.

```
#include <stdio.h>
#include <unistd.h>

int main(){
   pid_t pid = fork();
   if( pid < 0 ){
     fprintf(stderr, "Fork Failed");
     return 1;
}
   else if (pid==0) {
     pid_t pid1 = getpid();
     printf("child:pid=%d\n", pid); /*A*/
     printf("child:pid1=%d\n", pid1); /*B*/
}
  else{
    pid_t pid1 = getpid();
    printf("Parent:pid=%d\n", pid); /*C*/
    printf("Parent:pid1=%d\n", pid1); /*D*/
    wait(NULL);
}
return 0;
}
```

(C) What state information do you need to save/restore about threads when performing a context switch? Suppose a thread is running in a critical section of code, meaning that it has acquired the locks through proper arbitration. Can it get context switched? Why or why not?

(D) What are the possible outputs of this program ? Justify your answer.

```
#include <stdio.h>
#include <unistd.h>

void* threadfun(void* arg){
    print("hello\n");
}

int main(int argc, char** argv) {
    pthread_t t;
    pthread_create(&t, NULL, threadfun, NULL);
    sleep(1);
    printf("world\n");
}
```

(E) A recent popular game is having issues with its servers lagging heavily due to too many players being connected at a time. Below is the code that a player runs to play on a server:

```
void play_session(struct server s){
    connect(s);
    play();
    disconnect(s);
}
```

After testing, it turns out that the servers can run without lagging for a max of up to 1000 players concurrently connected. You are required to add **semaphores** to the above code to enforce a strict limit of 1000 players connected at a time? Assume that a game server can create semaphores and share them amongst the player threads. Explain how many sempahores are required and what you will initialize it to ?

(F) Some of the following arrangements of threads, and locks that they will grab, can lead to deadlock. For the scenarios given below, explain whether it is deadlocked or not.

(i) Thread 1 : will try to grab locks 1 and 2 in an arbitrary order
Thread 2 : will try to grab locks 1 and 2 in fixed order, 1 then 2.

(ii) Thread 1 : will try to grab locks 1 and 2 in fixed order, 1 then 2.
Thread 2 : will try to grab locks 1, 2 and 3 in fixed order, 1 then 2 then 3.

(iii) Thread 1 : will try to grab locks 1 and 2 in fixed order, 1 then 2.
Thread 2 : will try to grab locks 2 and 3 in fixed order, 2 then 3.
Thread 3 : will try to grab locks 1 and 2 in an arbitrary order

(iv) Thread 1 : will try to grab locks 1 and 2 in fixed order, 1 then 2.
Thread 2 : will try to grab locks 2 and 3 in fixed order, 2 then 3.
Thread 3 : will try to grab locks 1 and 3 in fixed order, 1 then 3.