1. Consider the following set of processes, with the length of the CPU-burst time given in milliseconds:

Process	Burst Time	Arrival time
*P*1	10	3
*P*2	1	0
*P*3	2	3
*P*4	3	4
*P*5	5	2

   a. Draw 3 Gantt charts illustrating the execution of these processes using FCFS, SJF, SRTF and RR (quantum = 2) scheduling.
   b. What is the turnaround time of each process for each of the scheduling algorithms in part a?

2. A computer provides each process with 32768 bytes of address space divided into pages of 4096 bytes.

   a. How many page frames could a process be allocated?

   b. A particular program has a text size of 11770 bytes, a data size of 10284 bytes, and a stack size of 9786 bytes. Will this program fit in the address space?

   c. If the page size were 512 bytes, would it fit?

Remember that a page may not contain parts of two different segments.

3. Given the following reference string:

0 2 1 3 0 1 4 0 1 2 3 4

   a. Show page faults occur during the processing of the reference scheme with LRU and FIFO, OPTIMAL, CLOCK? Each process is allocated 3 empty page frames.

   b. Estimate the ***hit ratio*** for LRU policies in a pure demand paging system if the TLB could contains 3 entries. Then compute the effective access time, suppose TLB lookup takes 5 nano sec, and memory access time is 100 nano sec.

4. Consider a system where the virtual memory page size is 2K (2048 bytes), and main memory consists of 4 page frames. Now consider a process which requires 8 pages of storage. At some point during its execution, the page table is as shown below:

Virtual page	Valid	Physical page
0 (0->2k-1)	No	
1(2k – 4k-1)	No	
2	Yes	1
3	No	
4	Yes	3
5	No	
6	Yes	0
7	Yes	2

a. List the virtual address ranges for each virtual page.

b. List the virtual address ranges that will result in a page fault

c. Give the main memory (physical) addresses for each of the following virtual addresses (all numbers decimal):

(i) 8500, (ii) 14000, (iii) 5000, (iv) 2100

c. List the virtual address ranges for each virtual page.

d. List the virtual address ranges that will result in a page fault.

e. Give the main memory (physical) addresses for each of the following virtual addresses (all numbers decimal): (i) 8500, (ii) 14000, (iii) 5000, (iv) 2100.

5. Given the following code:

```
1: int main(int argc, char *argv[]) {
3:          int a, e;
5:          a = 10;
6:          if (fork() == 0) {
7:                  a = a *2 ;
8:                  if (fork() == 0) {
9:                    a = a +1;
10:                   exit(2);
11:                 }
12:                 printf(" %d \n", a);
13:                 exit(1);
14:         }
15:         wait(&e);
16:         printf("a: %d;e : %d \n", a, WEXITSTATUS(e));
17:         return(0);
18:   }
```

Give the number of processes generated when running this program and the content display on the screen for each process.

a. new - ready - running - waiting - terminated
b. new - waiting - ready - running - terminated
c. new - ready - waiting - ready - running - terminated
d. new - ready - running - waiting - ready - running - terminated

6. There are two processes P0 and P1 in the system.
   The variable turn is set to 0. The code for process P0 is as follows:

   …..

   while (turn != 0) { } /* Do nothing and wait. */

   Critical Section /* . . . */

   turn = 0;

   ….

For *P1*, 0 is replaced by 1. Does this solution satisfy the mutual exclusion?

7. Using bit vector/linked list/grouping (n=4)/counting method for free space management, we assume that 3, 6, 8, 9, 10, 11, 12, 13, 17, 18, 19, 25, 26, 27 are empty blocks. How is free space managed?

Suppose that a disk drive has 2000 cylinders, numbered 0 to 1999. The drive is currently serving a request at cylinder 950, and the previous request was at cylinder 15. The queue of pending requests, in FIFO order, is: 86,1470, 913, 1774, 948, 1509, 1022,1750,130

8. Starting from the current head position, what is the total distance (number of cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms (SSTF, SCAN, C-SCAN, LOOK, C-LOOK) ?

9. Suppose a computer has a file system for a 256MB disk, where each disk block is 1MB.
   a. If the OS for this computer uses a FAT, what is the smallest amount of memory that could possibly be used for the FAT (assuming the entire FAT is in memory)? Explain.
   b. Given the following information:

File name	1st block
toto.txt	7

And the current content of FAT:

0	8
1	3
2	6
3	2
4	1
5	end-of-file
6	5
7	4
8	7
…..	…..

List all possible block for the file toto.txt.

10. Suppose that on a computer, the OS uses i-nodes and each disk block is 4KB. Assume that an i-node contains 12 direct block numbers (disk addresses) and the block numbers for one indirect block, one double indirect block, and one triple indirect block. Assume also that a block number is 4 bytes.

a. What is the largest possible file on that computer (assuming the disk is large enough).

b. On this computer, suppose you wanted to create a new file of the maximum size (that you computed for the answer a). How many free blocks on the disk would there need to be in order to create that file?

c. How many pointers of this i-node would be used in order to create the file toto.txt in the previous question?

11. Many CPU-scheduling algorithms are parameterized. For example, the RR algorithm requires a parameter to indicate the time slice. Multilevel feedback queues require parameters to define the number of queues, the scheduling algorithms for each queue, the criteria used to move processes between queues, and so on. These algorithms are thus really sets of algorithms (for example, the set of RR algorithms for all time slices, and so on). One set of algorithms may include another (for example, the FCFS algorithm is the RR algorithm with an infinite time quantum). What (if any) relation holds between the following pairs of algorithm sets?

    a. Priority and SJF

    b. Multilevel feedback queues and FCFS

    c. Priority and FCFS

    d. RR and SJF

12. Each process Pi, i = 0,1,2,3,……,9 is coded as follows.

```
repeat
 P(mutex)
 {Critical Section}
 V(mutex)
 forever
```

    The code for P10 is identical except that it uses V(mutex) instead of P(mutex). What is the largest number of processes that can be inside the critical section at any moment (the mutex being initialized to 1)?

13. Given the following code for P[i], i=0..3

```
wait (m[i]); wait(m[(i+1) mode 4]);

 //critical section

release (m[i]); release (m[(i+1)mod 4]);
```

When executing these processes, what will happen?