Name: ______________________________________ Student ID: ____________

Sample Mid-Term

The exam consists of two main parts:

- 20 true/false questions worth 40 points. For each of these questions:
  - a correct answer will get you 2 points
  - an incorrect answer will get you **0.5** negative points
  - a question left blank will get you 0 points

  So please **be careful** if you wish to use guesswork for questions in this part!
- <u>2 other questions worth 60 points (30 points each). There is NO negative grading for these questions.</u>

Name: ____ Student ID: ____________

Question 1 (40 points total) **Mark the following (T-F) questions using T for true, and F for false.**

1. Direct Memory Access (DMA) requires a special controller that facilitates the transfer of blocks between the I/O device and main memory.

2. The shell (or command interpreter) is part of the kernel.

3. An operating system can be viewed as a resource allocator to control various I/O devices and user programs.

4. An interrupt vector contains the saved program counter values of interrupted user programs.

5. Privileged instructions can be executed in monitor mode.

6. Priority Inversion is a condition that occurs in real-time systems where a lower priority process is starved because higher priority processes have gained hold of the CPU.

7. Multiprogramming is a technique used in an operating system for sharing a single processor between several independent jobs.

8. Soft real-time systems require that the scheduler firmly guarantee that critical processes will complete within a guaranteed amount of time.

9. Compiler/Linker is an example of a producer/consumer pair.

Name: ____ Student ID: _____________

10. The following code solves the critical section problem for two processes P_i and P_j.

```
Process Pi :
        repeat
                while turn != i do no-op;
                        critical section
                turn := j;
                        remainder section
        until false
```

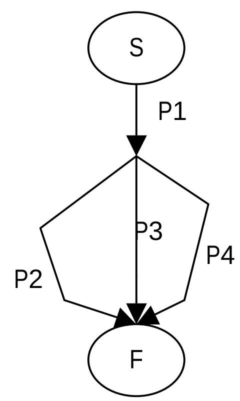
11. The definition of the wait and signal primitives in a semaphore are as follows:
The semaphore(S) is initialized to 1.

wait(S): **while** S > 0 do no-op;
S := S - 1;

signal(S); S := S + 1

12. When a child process is created using the **fork()** system call, the child process inherits all the open file descriptors of the parent process.

13. Do the semaphore operations below correctly describe the process flow graph in the Figure.



p1: body; V(s1); V(s1);
p2: P(s1); body;
p3: P(s1); body;
p4: P(s1); body;

14. In synchronous I/O, the I/O call returns quickly with as much data as available.

15. In Solaris 2 the LWPs (Light Weight Processes) are the kernel threads.

16. In Multi-level feedback queue scheduling processes are **not** allowed to move from one queue to another.

17. The *virtual machine* is the logical conclusion in the layered approach to designing operating systems.

18. Maskable interrupts are those that can be ignored.

19. Symmetric multiprocessing treats all processors as equals, and I/O can be processed on any CPU.

20. Thread scheduling overhead is higher for user threads than kernel threads.

Name: ______________________________________ Student ID: ____________

Question 2 (30 points) Processes/Threads & CPU Scheduling

(a) What resources are used when a thread is created? How do these differ from those used when a process is created? (no more than a few sentences) **(4 points)**

(b) What are context switches used for and what does a typical context switch involve? (no more than a few sentences) **(4 points)**

(c) Consider the following set of processes:

Process ID	Arrival Time	Priority	Burst Time
A	2	3	100
B	6	1 (highest)	40
C	10	2	80
D	16	4 (lowest)	20

Name: ____ Student ID: ____________

(c1) Draw Gantt charts illustrating the execution of these processes using the following algorithms.

- First Come First Serve **(3 points)**

- Shortest Remaining Time First (Preemptive) **(4 points)**

- Round-Robin (quantum = 20) **(3 points)**

(c2) Based on your work above, fill in the table below giving both the waiting time and finish time for each process **(12 points)**

Waiting Time

Process ID	**FCFS**	**SJF (P)**	**Round Robin**
A			
B			
C			
D			

Finish Time

Process ID	**FCFSS**	**SJF (P)**	**Round Robin**
A			
B			
C			
D			

Name: ____ Student ID: ____________

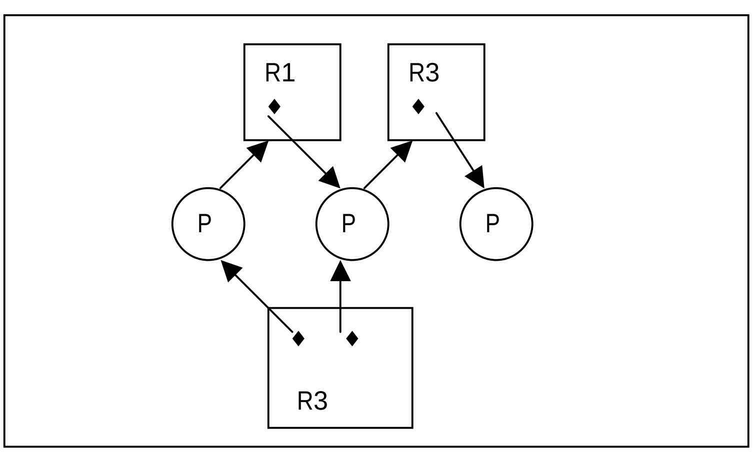
Question 3 **(30 points) Process Synchronization, Critical Sections & Deadlocks**

(3a) What is the critical section problem? List and state the three requirements that a solution to the critical section problem must satisfy. **(5 points)**

(3b) List the four conditions that must hold simultaneously in a system for deadlock to arise. **(4 points)**

(3c) In the following resource allocation graph, is the state a deadlocked one? If so which ones are deadlocked? **(3 points)**

Resource allocation graph. R_i = Resource, P = process.



Name: ____ Student ID: ____________

(3d) *Cigarette-Smokers Problem.* Consider the following problem (you may remember this from Homework 1) **(10 points)**

Consider a system with three smoker processes and one agent process. Each smoker continuously rolls a cigarette and then smokes it. But to roll and smoke a cigarette, the smoker needs three ingredients: tobacco, paper and matches. One of the smoker processes has paper, another has tobacco and the third has the matches. The agent has an infinite supply of all three materials. The agent places two of the ingredients on the table. The smoker who has the remaining ingredient then makes and smokes a cigarette, signaling the agent on completion. The agent then puts out another two of the three ingredients, and the cycle repeats.

Given below is a solution to the Cigarette-Smokers Problem. Give initial conditions for the semaphores as well as plausible values for the variables i & j and r & s, such that the agent and smokers are synchronized. Write a couple of sentences on why these initial conditions are necessary and sufficient.

Solution:

```
var a: array [0..2] of semaphore {initial condition =    }
agent: semaphore {initial condition =    }
```

agent code

```
rep
repeat
  Set i,j to a value =
  wait(agent);
  signal(a[i]);
  signal(a[j]);
until false;
```

smoker code

```
rep
repeat
  Set r,s to a value =
  wait(a[r]);
  wait(a[s]);
  "smoke"
  signal(agent);
until false;
```

Explanation (brief):

(3e) *Test-and-Set.* The Test-and-Set instruction is used in hardware to achieve synchronization. It can be defined in the following way:

```
function Test-and-Set (var target: boolean): boolean;
   begin
        Test-and-Set := target;
        target := true;
   end;
```

Now show how the Test-and-Set instruction can be used to protect a critical region and hence achieve mutual exclusion (do not worry about satisfying the bounded waiting condition).
(8 points)

```
repeat





until false;
```

END OF PAPER.