Tracing the behavior of two mutually excluded concurrent processes:

This hw is to be done strictly independently.

You have to write a program which does the following:

(A) Given a command
semaphore S(p)
where *semaphore* is a keyword, *S* is the name of the semaphore, and *p* is an integer constant used as the initial value of the semaphore S, it will set up the semaphore S, initialing and setting up the associated waiting queue.

(B) Given a command
wait(S)
where *wait* is a keyword, and *S* is a semaphore, it will implement the semantics of *wait* (as explained in class).

(C) Given a command
signal(S)
where *signal* is a keyword, and *S* is a semaphore, it will implement the semantics of *signal* (as explained in class).

(D) Now, finally your program will *trace* the execution of any *arbitrary*, but *valid*, sequence (see discussions in class) by two mutually excluded concurrent processes.

- Your program will be evaluated based on the usual metrics (see the model software given in class)
- Your output must display
  - The value and the status of the semaphore queue;
  - the instructions being executed at each cycle;
  - and annotations explaining the event that arises in each cycle – meaning why a particular command in the arbitrary sequence is "not valid" and hence ignored.