Task Series 2

2.1 Binary semaphores

Implement your own "library" that provides binary semaphores with the functions:

int mysem_init(mysem_t *s, int n);	Initialize semaphore with value n. Returns 1 on success, 0 if n ≠ 0.1, and -1 if the semaphore is already initialized.
int mysem_down(mysem_t *s);	Decrement semaphore by 1. Returns 1 on success or -1 if semaphore is uninitialized.
int mysem_up(mysem_t *s);	Increment semaphore by 1. Returns 1 on success, 0 if semaphore is already 1, or -1 if semaphore is uninitialized.
int mysem_destroy(mysem_t *s);	It destroys the traffic light. Returns 1 on success or -1 if the semaphore is uninitialized / already destroyed.

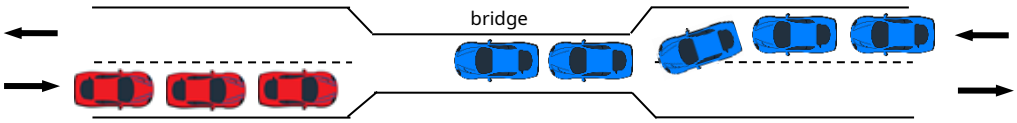
Your implementation must (internally) use the system V semaphores (assume they are fair).

2.2 Recognition of prime numbers

Change the code you developed for task 1.2 so that all desired synchronization is implemented with your own semaphores.

2.3 Narrow bridge

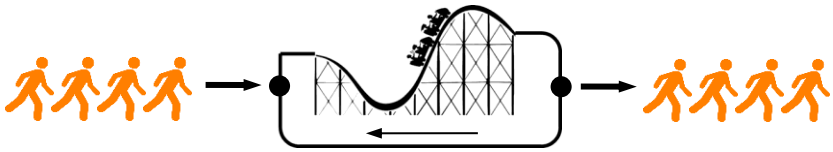
Develop control code to regulate vehicular traffic over a bridge so that the following properties are ensured: (1) No vehicles moving in opposite directions are allowed on the bridge. (2) No more than N vehicles are allowed on the bridge. (3) A vehicle is not allowed to wait forever to cross the bridge, even if vehicles are constantly arriving on the other side.



Implement synchronization between vehicles/threads, via appropriate "entry"/"exit" code that each vehicle executes when it arrives and after crossing the bridge, respectively. The time to cross the bridge is simulated through artificial waiting. Synchronization must be done without any thread acting as a mediator/traffic controller. Your implementation should be based on your own binary semaphores. Test/ demonstrate your solution through onesimple simulation program that creates vehicles on each side of the bridge at specific times based on information it reads from its input.

2.4 Roller coaster

An amusement park roller coaster holds N passengers. The roller coaster only starts its next ride when it is full, and passengers disembark from the roller coaster after it has completed the current ride, and before the next passengers start boarding.



Implement the desired synchronization between the passengers and the roller coaster, without any other auxiliary thread. Your implementation should be based on your own binary semaphores. Test/demonstrate your solution through onesimple simulation program that initially creates a thread for the roller coaster and then additional passenger threads at specific times based on information it reads from its input. The number of passengers the roller coaster can hold is given as an argument to the program. The time it takes the roller coaster to complete the route is simulated through artificial waiting.

The implementation must be done in C using the pthreads library. Synchronization in tasks 2.2-2.4 must be implemented exclusively with**your own binary semaphores**and**with no active standby**.

Tradition: SaturdayDecember 9, 2023, 11:59 p.m