A) Para poder asegurar que la serie pueda ser ejecutada es necesario un mutex, también está el caso de que si fuera un semáforo se podría tener a las competidores en paralelo, pero podría dar problemas si en algún caso se mete otro competidor en la serie porque alguno terminó previamente, lo cual no puede suceder.

- Entonces se define este mutex que toma la serie como un proceso completo asegurando que una serie no comenzará a menos que termine la que esté en proceso.

```
def mutex(i, k, mutex):
    mutex.acquire(1)
    time.sleep(tiempo_total_carrera)
    i = i - k
   print(resultados(random))
    mutex.release()
    return i

cantidad_participantes_total = 27
participan = 9
series=27/9

mutex = Lock()
for i in range(0, series):
        cantidad_participantes_total= mutex(cantidad_participantes_total, participan, mutex)
```

B) Se podría utilizar first come first served o cualquier algoritmo que no tome procesos y los pare en algún momento, ya que al ser una carrera/serie no es algo que se pueda detener para seguir después.

- En el caso hipotético de que se pudiera detener la carrera y continuar después, se podrían tomar los otros algoritmos dependiendo la necesidad que exista.