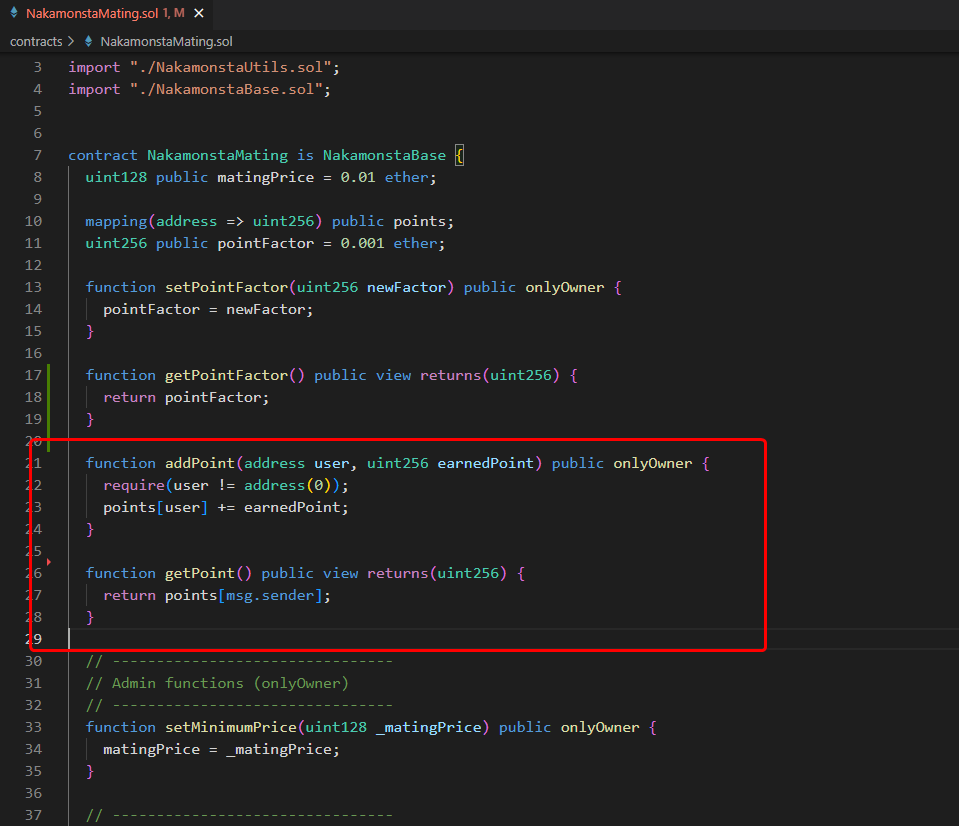
Write a Rust program that asynchronously monitors the contract in a non-blocking loop. Hint: you can either use OS threads or green threads in a runtime like Tokio. When the state of your contract satisfies a constraint, take an action (like printing the satisfied state) and close the program (including properly handling the shutdown of your parallel threads).

For example, the main thread of your program should deploy and set up the contract. Spawn a parallel thread to monitor for a certain condition. Back in the main thread, take an action to satisfy the condition. Then acknowledge to the user that the condition was met, reset the contract state, and shut down the program.

Prepare:

Deployed a smart contract on Ganache testnet.

It contains two functions – addPoint and getPoint



Main thread

Add 100 points to test account after sleeping some time.

And wait until child thread is finished

Child thread

Check point of the test account until it has positive point

Both threads print log to check performance. You can check result in the screen capture video.