🎄 Christmas Rush: Build a Mini Fail2Ban Module

Scenario:
It's a cold December day, just three days before Christmas. Your boss, **Michelle Scott**, has handed you a last-minute task critical to the security of your company's online platform. With heightened traffic and the holidays fast approaching, malicious activity is on the rise.

Your teammate **Geno Linetti** is building the logic that will respond to detected threats. **You are responsible for building a detection module** that scans application logs, identifies suspicious IPs, and blocks them.

To help out, your colleague **Jill** has recommended using the IPLocate API to enrich your IP detection logic—especially to flag known **Tor exit nodes** or **traffic from China**, which should be blocked automatically.

🧠 Your Task

Create a **Python script** that mimics Fail2Ban-style behavior:

- Accepts one or more application log files via command-line arguments
- Detects abusive or high-risk IPs
- Simulates blocking those IPs with printed output

💡 Technical Requirements

Your script must:

1. **Accept log files** as command-line arguments:

```shell
python block_ips.py app1.log app2.log
```

2. **Parse each line** to extract:

- IP address

3. Example line:

```
None
2025-12-22 15:12:07 [WARN] Failed login from 192.168.1.10
```

4. **Blocking Logic:**

   Your script must block IPs under the following conditions:

   ✅ **Always block** if:

   - The IP is identified as coming from **Tor**
   - The IP is located in **China, Russia, or North Korea**

5. **Simulate blocking** by printing:

```
None
[BLOCKED] IP 192.168.1.10 has been blocked due to suspicious activity.
```

6. Ensure **no duplicate blocks** are reported for the same IP.

🧪 Sample Usage

```
Shell
python block_ips.py logs/app.log
```

Example Output:

```
None
[BLOCKED] IP 203.0.113.42 (China, Tor Exit Node) has been blocked.
[BLOCKED] IP 198.51.100.17 has been blocked due to suspicious activity.
```

📦 Deliverables

- A single `block_ips.py` script
- Inline comments explaining your logic