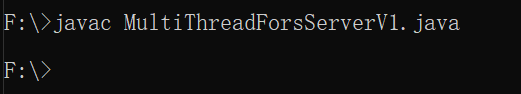
设计思路：

V1 服务端启动n 个线程（由参数提供）， 每个线程需要完成对S序列的子序列的计算任务。因此定义一个计算任务CalculateS 实现 Runnable 接口，在计算任务CalculateS 中，分别定义了square\pow\... 对序列的计算过程， 根据指定的操作类型，进行相应的计算逻辑。V1分别将客户端传来的S序列平均分配到n 个线程去处理。

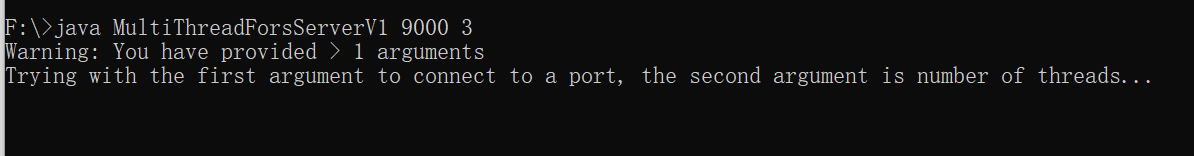
V2中定义了与V1相同的计算任务CalculateS, 与V1 不同的是S序列的分配方式，V2中前n-1个线程每个线程均处理只含一个number的序列，最后一个线程处理其余子序列。

## V1

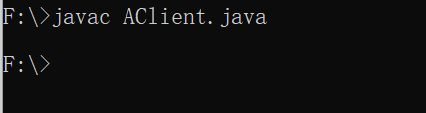
Compile server



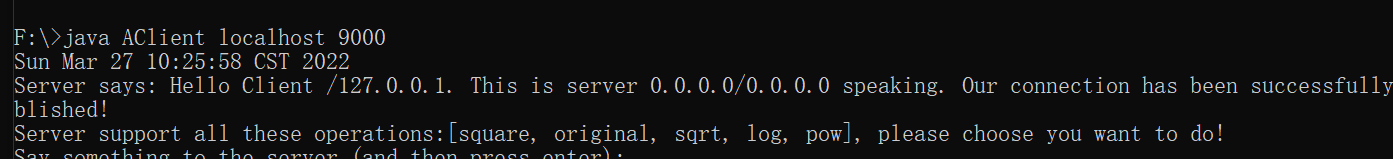
Execute server



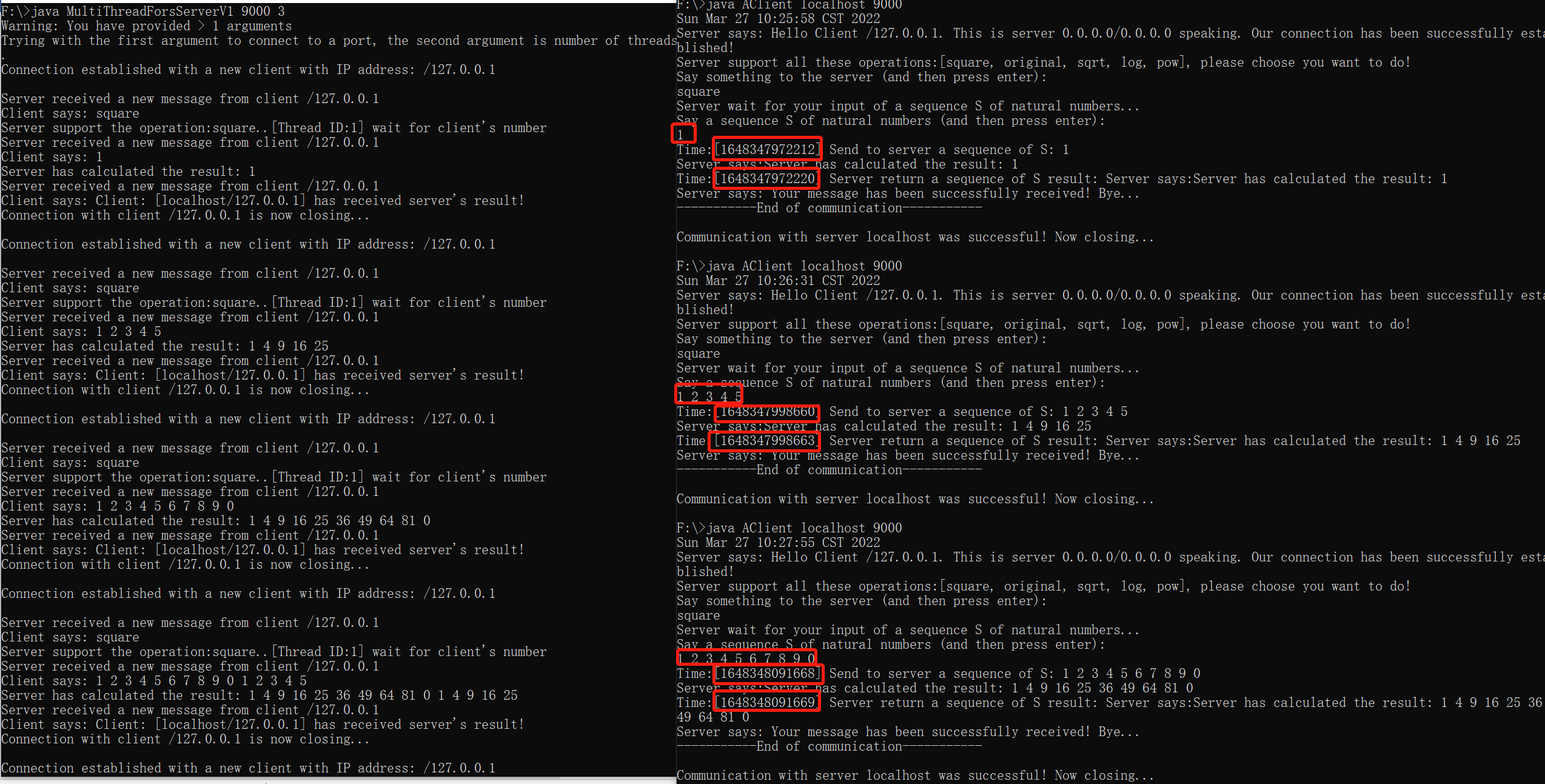
Compile client

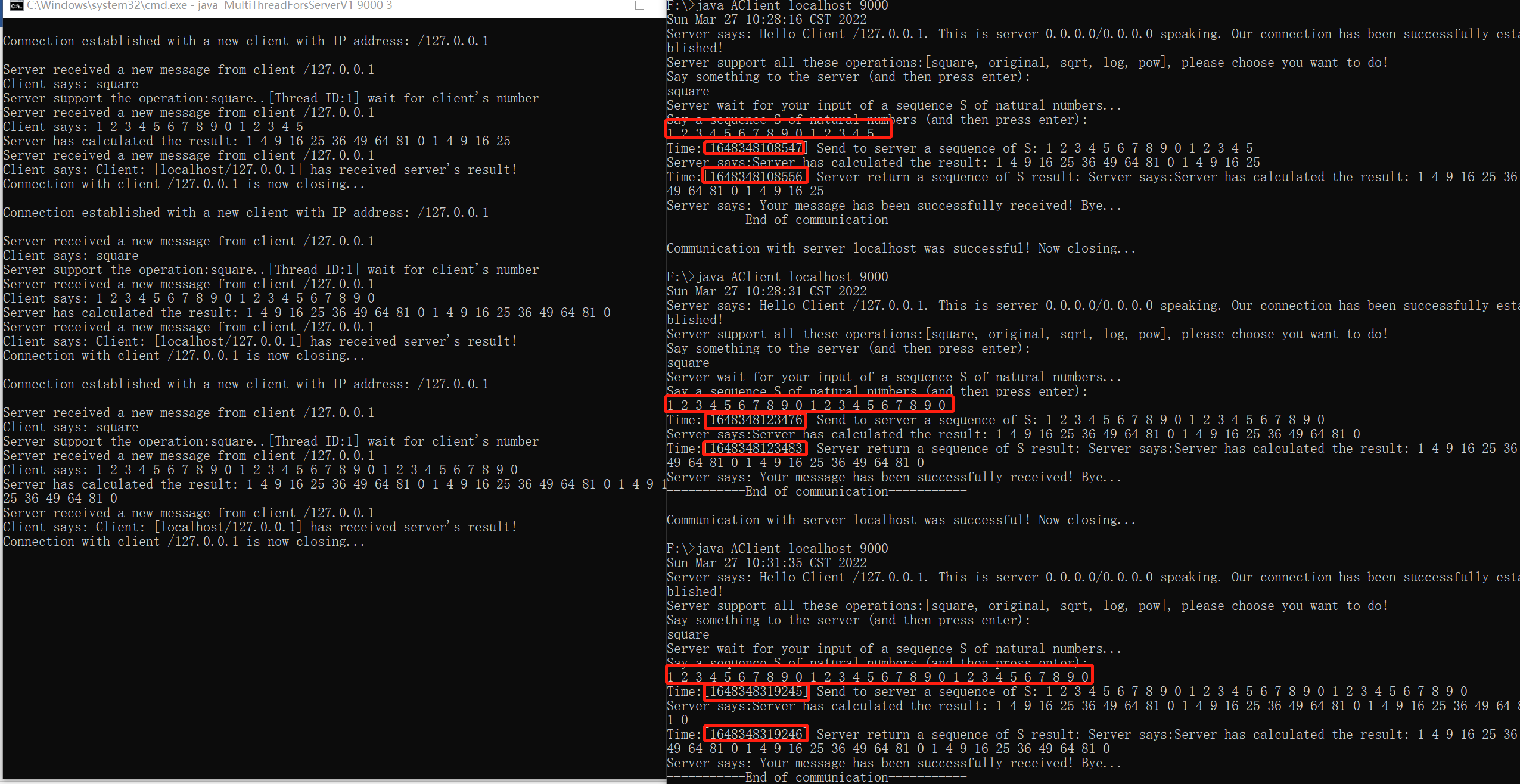


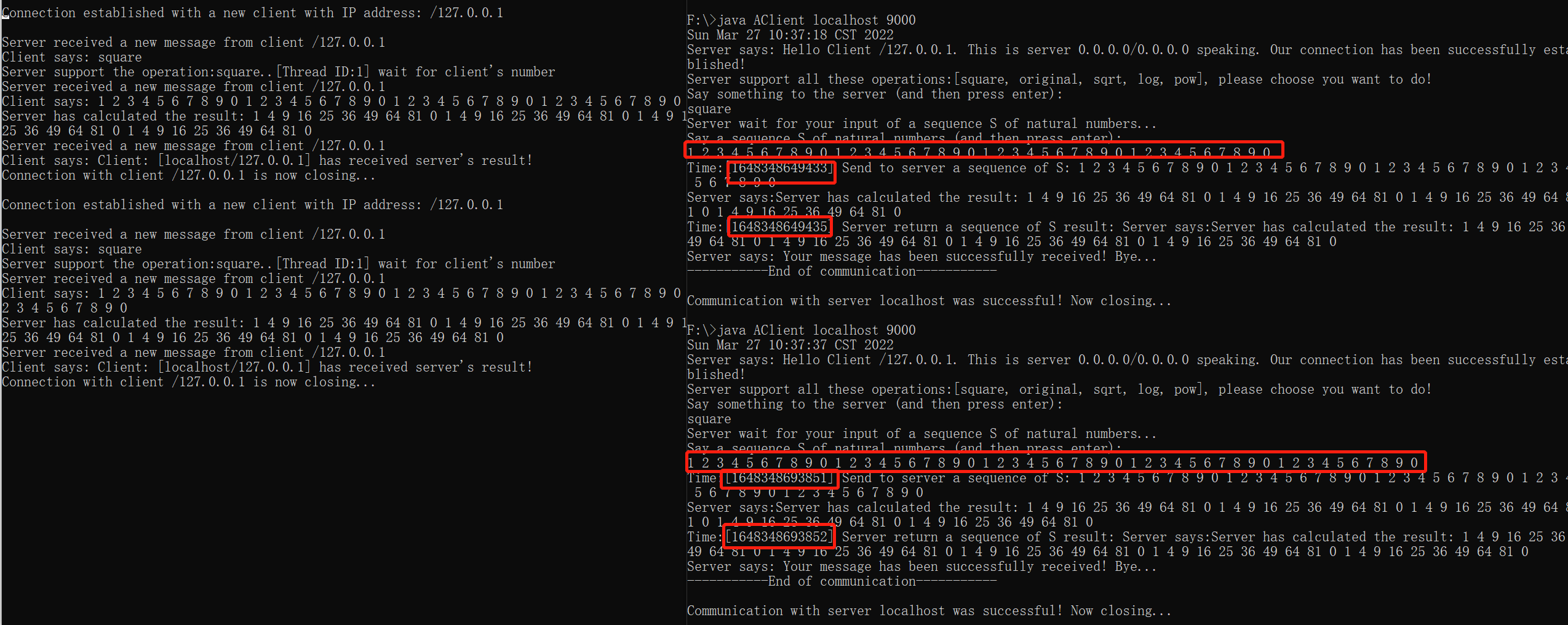
Execute client



Execution examples:

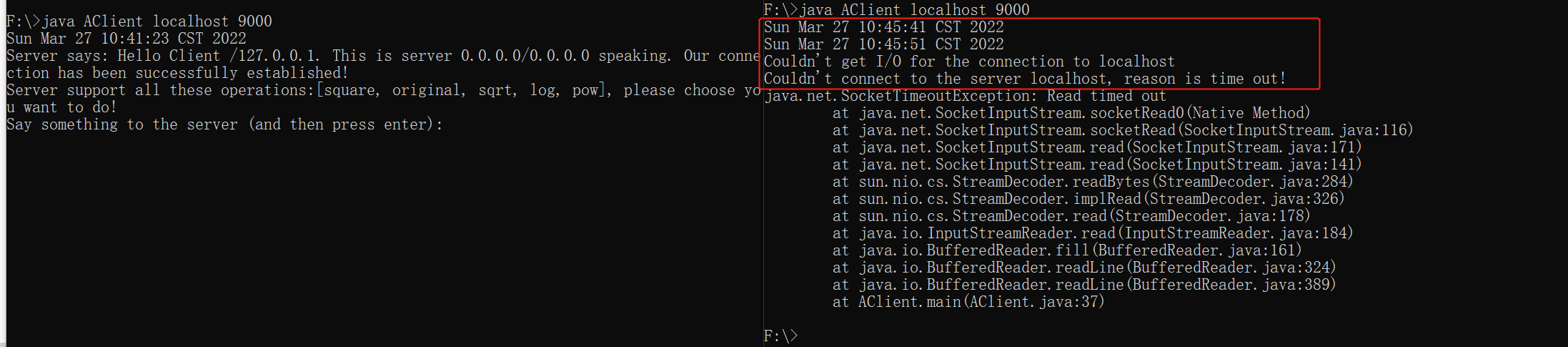






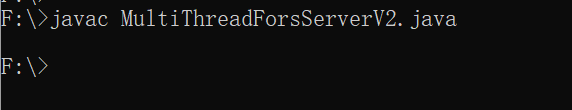
|  |  |  |
| --- | --- | --- |
| client input | server threads | time consumed(ms) |
| 1 | 3 | 8 |
| 1 2 3 4 5 | 3 | 3 |
| 1 2 3 4 5 6 7 8 9 0 | 3 | 1 |
| 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 | 3 | 9 |
| 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 | 3 | 7 |
| 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 | 3 | 1 |
| 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 | 3 | 2 |
| 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 | 3 | 1 |

When another client want to connect to the server:

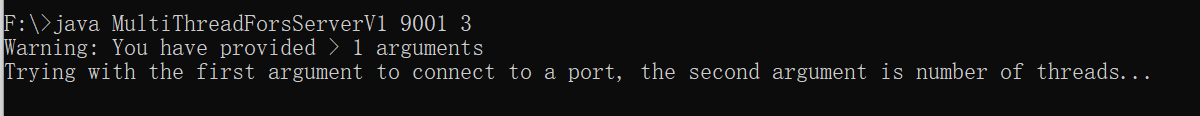


## V2

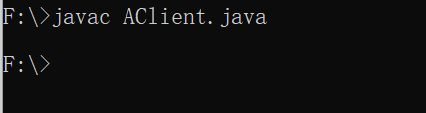
Compile server



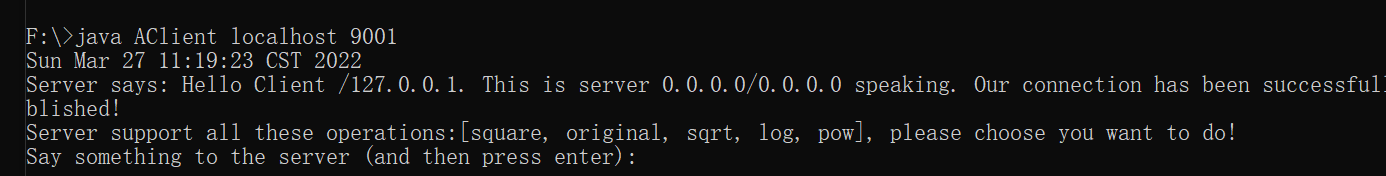
Execute server



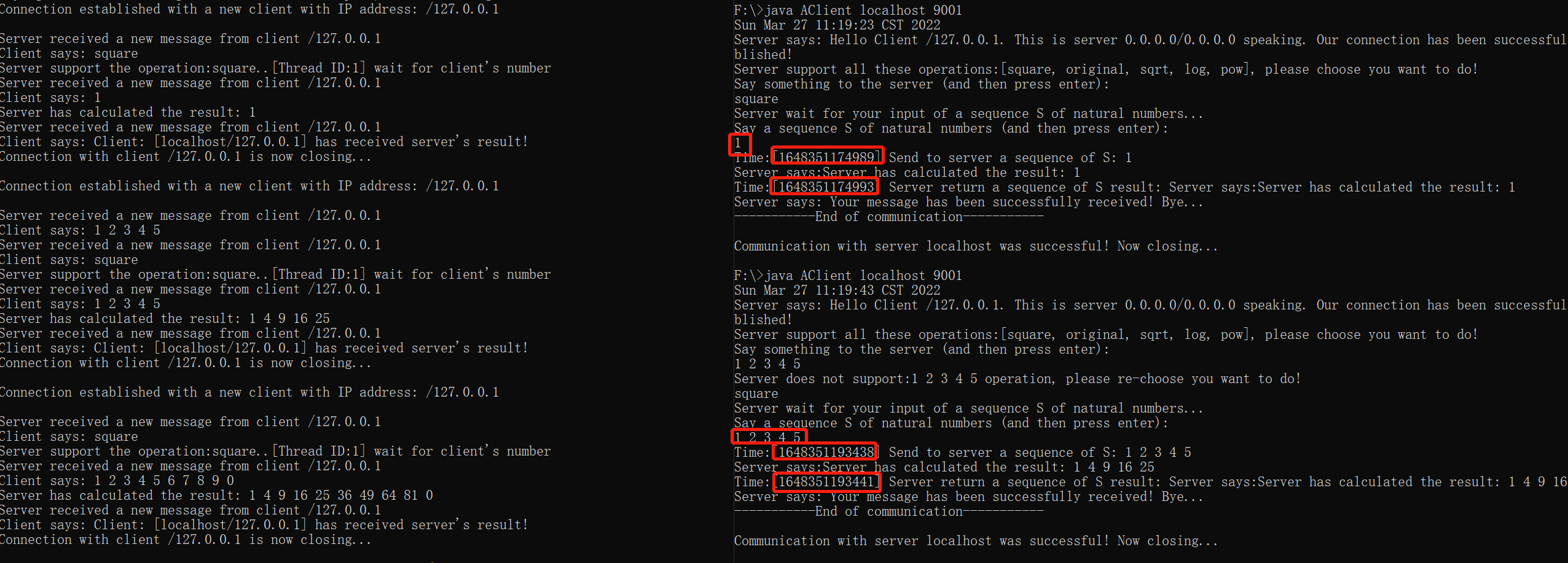
Compile client

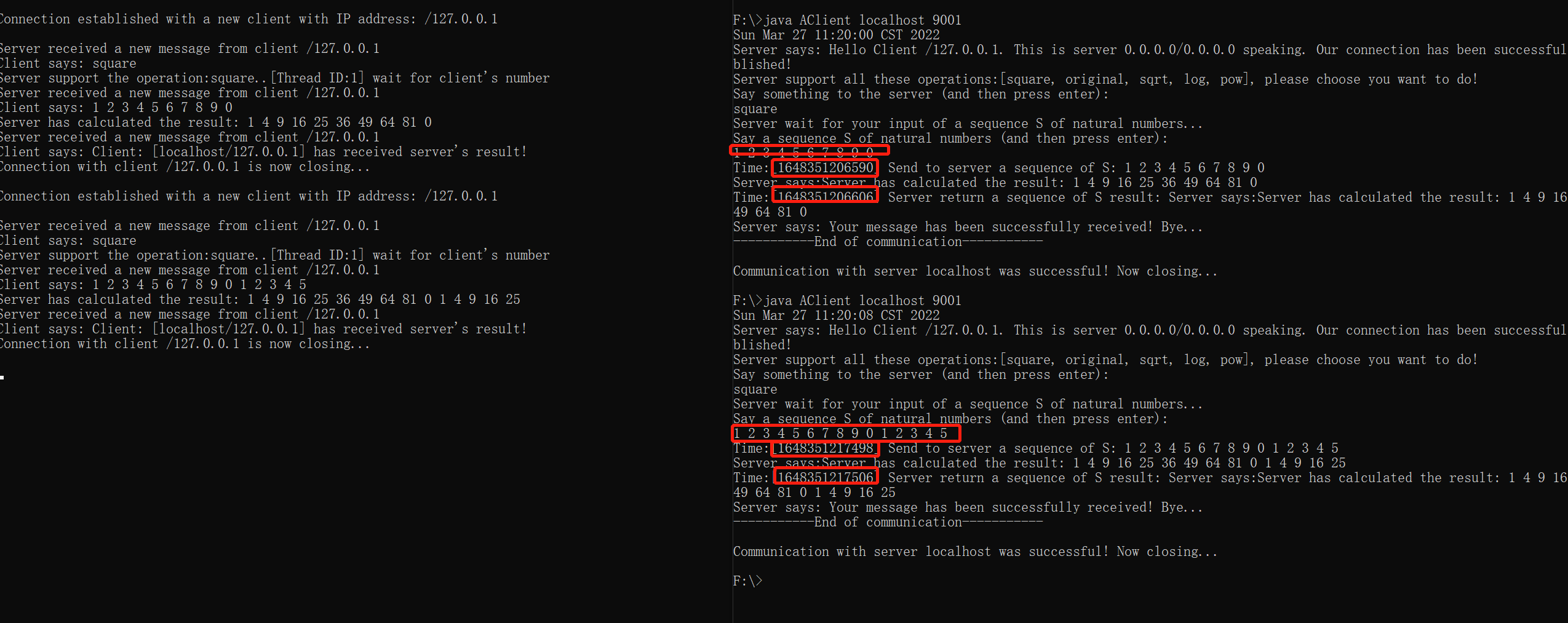


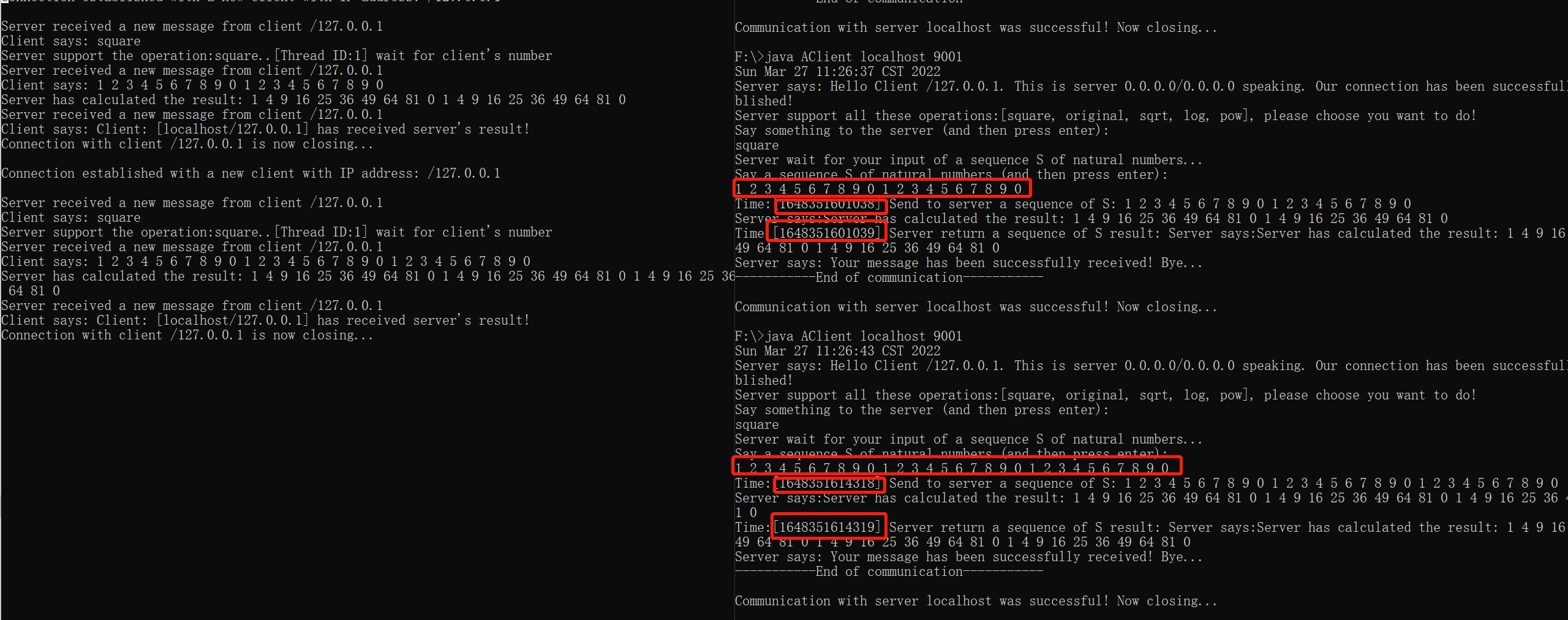
Execute client

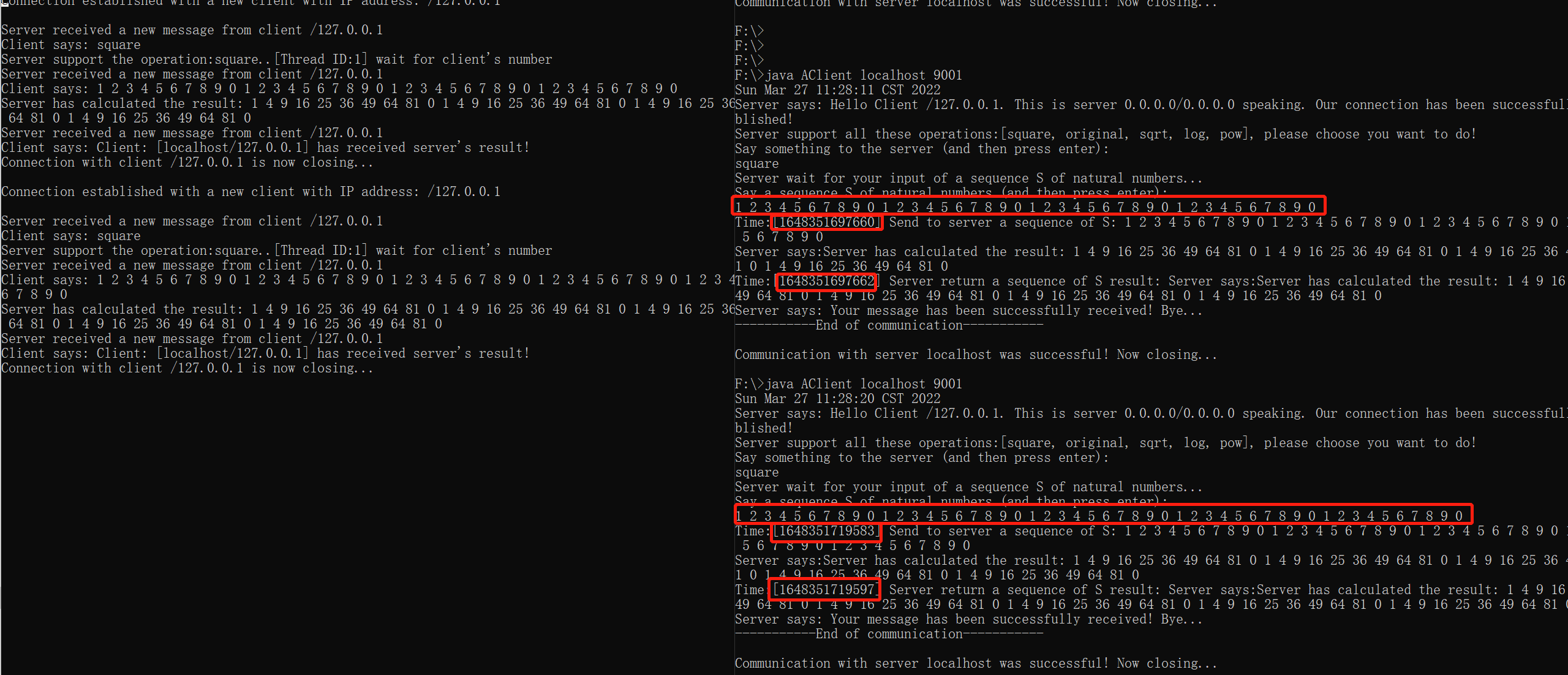


Execution examples:









|  |  |  |
| --- | --- | --- |
| client input | server threads | time consumed(ms) |
| 1 | 3 | 4 |
| 1 2 3 4 5 | 3 | 3 |
| 1 2 3 4 5 6 7 8 9 0 | 3 | 16 |
| 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 | 3 | 8 |
| 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 | 3 | 1 |
| 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 | 3 | 1 |
| 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 | 3 | 2 |
| 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 | 3 | 14 |