1. What two languages do they consider?
*Java
*GForth
Python
Javascrip
Typescript
Lua
C++

2. How does replication (as defined in the paper) make interpreters faster?
*Makes indirect jumps (branches) more predictable thereby reducing the cost per dispatch
Reduces the number of dispatches
Makes fat bytecodes thin
Reduces the number of handlers

3. How does the use of superinstructions (as defined in the paper) make interpreters faster?
Makes indirect jumps (branches) more predictable thereby reducing the cost per dispatch
*Reduces the number of dispatches
Makes fat bytecodes thin
Reduces the number of handlers

4. Which language does the paper say the optimizations perform worse for?
*Java
GForth
Python
Javascrip
Typescript
Lua
C++

5. Which has the fewest indirect branches (or should in general)?
switch-dispatch interpreter
decode-and-dispatch interpreter
*direct threaded interpreter
indirect threaded interpreter