Here is a Java command - and YES it is just ONE command

```java
{ 
    x=3;
    y=cin.nextInt();
}
```

NOTICE: the semicolons ( ; ) are PART of the **atomic** commands that are leaves in the tree.
Although you may be able to easily draw the trees for simple commands - the homeworks and exam questions will provide you with more challenging examples.

There is a METHOD to guarantee you get the CORRECT answer.

However, it MUST BE FOLLOWED exactly.

The most important rule in the METHOD is:

**NEVER SKIP AHEAD.**
1. Use TWO sheets of your own paper
2. Copy the Java steps onto ONE of the sheets - do NOT CROWD your copy.
3. Your TREE will go on the OTHER sheet of paper.
4. REPEATEDLY DO THESE STEPS IN ORDER:
   a. Draw a STUB in your Tree
   b. UNDERLINE a piece:
      i. A conjunction, or
      ii. A parenthesized expression, or
      iii. An atomic step.
   c. DRAW the underlined piece within the TREE.

There will be a few exceptions to this pattern.
HOWEVER - **MEMORIZE**:

STUB - UNDERLINE - DRAW - repeat
if ( x<3 ) {
    X = 5;
    y = y+2;
}
Java Command

if ( x<3 ) {
    X = 5;
    y = y+2;
}

Command Tree
if ( x<3 ) {
    X = 5;
    y = y+2;
}

Draw the conjunction - the one just underlined
if ( x<3 ) {
    X = 5;
    y = y+2;
}
```java
if ( x<3 ) {
    X = 5;
    y = y+2;
}
```
Java Command

```java
if (x<3) {
    X = 5;
    y = y+2;
}
```

Command Tree

```
if
```

**NOTE:** we leave the parentheses OUT when drawing IN the TREE.

Draw the underlined piece - always under the newest STUB. In this case, the drawn piece is a LEAF.
Java Command

```java
if ( x<3 ) {
    X = 5;
    y = y+2;
}
```

Since an if has at least **two** legs, we add another stub to the if node.
Java Command

```java
if ( x<3 ) {
    x = 5;
    y = y+2;
}
```

The underlined CONJUNCTION-Called SEQ

NOTE: we call the conjunction SEQ but code it using a pair of braces `{ }`
HOWEVER, only underline the FIRST brace at this time.
Java Command

```
if ( x < 3 ) {
    x = 5;
    y = y + 2;
}
```

Command Tree

```
if

x < 3

seq

DRAW

conjunction
```
Java Steps

```java
if ( x<3 ) {
    x = 5;
    y = y+2;
}
```
Java Steps

```
if ( x < 3 ) {
  x = 5;
  y = y + 2;
}
```

Command Tree

- `if`
  - `x < 3`
  - `seq`
Java Steps

```
if ( x<3 ) {
    x = 5;
    y = y+2;
}
```

Command Tree

```
if
  x<3
  seq
  x=5; ← DRAW
```
if ( x<3 ) {
    x = 5;
    y = y+2;
}

NOTE: A SEQ node can have any number of legs - keep adding STUBS to the SEQ until the close brace } gets underlined.
Java Command

```java
if (x<3) {
  x = 5;
  y = y+2;
}
```

Two of the basic steps have been drawn on the same page.

(1) UNDERLINE

Command Tree

(2) DRAW

if

x<3

seq

x=5;

y=y+2;
Java Steps

WHEN a close brace } is detected next to be underlined - the work flow alters to:
UNDERLINE, CHECK SEQ NODE

```
if ( x<3 ) {
    x = 5;
    y = y+2;
}
```

(1) UNDERLINE

(2) CHECK

NOTE: checking a node means that NO MORE legs can be added to it.
Java Steps

```java
if ( x<3 ) {
    x = 5;
    y = y+2;
}
```

WHEN a Node has enough "LEGS" the work flow changes to:

CHECK

NOTE: the node checking proceeds back UP the tree until a node is found that can have another leg.
Java Steps

```
if ( x<3 ) {
    x = 5;
    y = y+2;
}
```

WHEN FINISHED:
(1) ALL NODES ARE CHECKED
(2) ALL CODE IS UNDERLINED

Any Extra code or Unchecked Nodes means there is a mistake.
To know when a Node should be checked it is necessary to know how many legs a Node may have.

**The rules are VERY STRICT**

1. **ALL NODES HAVE TWO LEGS**
2. **EXCEPT If-Else which has exactly THREE LEGS**
3. **EXCEPT Seq which can have any number of LEGS.**