Homework 22

1.

A diagram of a computer

Description automatically generatedA diagram of a call

Description automatically generatedA screenshot of a computer screen

Description automatically generatedA diagram of a block

Description automatically generatedA diagram of a block

Description automatically generated

2.

/\*\*

\* Reports the number of calls to primitive instructions (move, turnleft,

\* turnright, infect, skip) in a given {@code Statement}.

\*

\* **@param** s

\* the {@code Statement}

\* **@return** the number of calls to primitive instructions in {@code s}

\* **@ensures** <pre>

\* countOfPrimitiveCalls =

\* [number of calls to primitive instructions in s]

\* </pre>

\*/

**public** **static** **int** countOfPrimitiveCalls(Statement s) {

**int** count = 0;

**switch** (s.kind()) {

**case** ***BLOCK***: {

/\*

\* Add up the number of calls to primitive instructions

\* in each nested statement in the BLOCK.

\*/

**for** (**int** i=0; i < s.lengthOfBlock(); i++) {

count += *countOfPrimitiveCalls*(s.removeFromBlock(i));

}

**break**;

}

**case** ***IF***: {

/\*

\* Find the number of calls to primitive instructions in

\* the body of the IF.

\*/

**for** (**int** i=0; i < s.lengthOfBlock(); i++) {

count += *countOfPrimitiveCalls*(s.removeFromBlock(i));

}

**break**;

}

**case** ***IF\_ELSE***: {

/\*

\* Add up the number of calls to primitive instructions in

\* the "then" and "else" bodies of the IF\_ELSE.

\*/

**for** (**int** i=0; i < s.lengthOfBlock(); i++) {

count += *countOfPrimitiveCalls*(s.removeFromBlock(i));

}

**break**;

}

**case** ***WHILE***: {

/\*

\* Find the number of calls to primitive instructions in

\* the body of the WHILE.

\*/

**for** (**int** i=0; i < s.lengthOfBlock(); i++) {

count += *countOfPrimitiveCalls*(s.removeFromBlock(i));

}

**break**;

}

**case** ***CALL***: {

/\*

\* This is a leaf: the count can only be 1 or 0. Determine

\* whether this is a call to a primitive instruction or not.

\*/

**if** (s.kind().equals(IDENTIFIER)) {

count++;

}

**break**;

}

**default**: {

// this will never happen...can you explain why?

// because all possible results are already addressed

**break**;

}

}

**return** count;

}