public static void simplifyIfElse(Statement s) {

switch (s.kind()) {

case BLOCK: {

int length = s.lengthOfBlock();

for (int i = 0; i < length; i++) {

Statement subStatement = s.removeFromBlock(i);

simplifyIfElse(subStatement);

s.addToBlock(i, subStatement);

}

break;

}

case IF: {

Statement subStatement = s.newInstance();

Statement.Condition condition = s.disassembleIf(subStatement);

simplifyIfElse(subStatement);

s.assembleIf(condition, subStatement);

break;

}

case IF\_ELSE: {

Statement subStatementIf = s.newInstance();

Statement subStatementElse = s.newInstance();

Statement.Condition condition = s.disassembleIfElse(subStatementIf, subStatementElse);

switch (condition) {

case NEXT\_IS\_NOT\_EMPTY:

condition = Statement.Condition.NEXT\_IS\_EMPTY;

break;

case NEXT\_IS\_NOT\_ENEMY:

condition = Statement.Condition.NEXT\_IS\_ENEMY;

break;

case NEXT\_IS\_NOT\_FRIEND:

condition = Statement.Condition.NEXT\_IS\_FRIEND;

break;

case NEXT\_IS\_NOT\_WALL:

condition = Statement.Condition.NEXT\_IS\_WALL;

break;

default:

simplifyIfElse(subStatementIf);

simplifyIfElse(subStatementElse);

s.assembleIfElse(condition, subStatementIf, subStatementElse);

return;

}

simplifyIfElse(subStatementIf);

simplifyIfElse(subStatementElse);

s.assembleIfElse(condition, subStatementElse, subStatementIf);

break;

}

case WHILE: {

Statement subStatement = s.newInstance();

Statement.Condition condition = s.disassembleWhile(subStatement);

simplifyIfElse(subStatement);

s.assembleWhile(condition, subStatement);

break;

}

case CALL: {

// No action needed for CALL statements because they do not contain any nested statements to process

break;

}

default: {

// This case will never occur because all kinds of statements are covered by the above cases

break;

}

}

}