```
/*
A magma program for solving sudoku problems. See
http://en.wikipedia.org/wiki/Sudoku
for information about sudokus.
*/

Z:=IntegerRing();
J:=[1..9];
U:=Set(J);

Sn:=func<i,j|3*(Ceiling(i/3)-1)+Ceiling(j/3)>; //square number

sol:=func<G |
forall{i : i in J | {A[i,j] : j in J} eq U} and
forall{j : j in J | {A[i,j] : i in J} eq U} and
forall{n : n in J | {A[i,j] : i,j in J | Sn(i,j) eq n} eq U}
where A is Matrix(Z,9,9,G)>;  //G is a solution

Rd:=func<G,i| U diff {A[i,j] : j in J | A[i,j] ne 0} where A is
Matrix(Z,9,9,G)>;  //Rd(G,i) is the set of numbers missing from row i

Cd:=func<G,j| U diff {A[i,j] : i in J | A[i,j] ne 0} where A is
Matrix(Z,9,9,G)>; //Rd(G,j) is the set of numbers missing from column j

Sd:=func<G,n| U diff {A[i,j] : i,j in J | A[i,j] ne 0 and Sn(i,j) eq n}
where A is Matrix(Z,9,9,G)>;
//Sd(G,n) is the set of numbers missing from square n

poss:=function(G,i,j);
return Rd(G,i) meet Cd(G,j) meet Sd(G,Sn(i,j));
end function;
/* poss(G,i,j) is the set of numbers missing from the row, column, and
square of (i,j) */

Bm:=func<G|
Matrix(Z,9,9,[<i,j,A[i,j] eq 0 select #poss(G,i,j) else -1> : i,j in
J]) where A is Matrix(Z,9,9,G)>;
/* Bm(G) gives the number of  numbers missing from the row, column, and
square of (i,j). If entry (i,j) has been filled out, we assign -1 to it.
*/



/* Function "reduction" applies two principles:
1. If in entry (i,j) there is only one number missing from the row, column, and
square of (i,j) then write that number as entry (i,j)
2. If k is missing from row i, find the places in row i that may
contain k. If there is only one such place in row i, write k in that
place. Similarly for columns and squares.
The function applies these two principles repeatedly until it can not
fill out more entries in the matrix.
*/
reduction:=function(G);
B:=Bm(G);
g:=#G; d:=1;
```

```
repeat
//Principle 1
while 1 in {B[i,j] : i,j in J} do
G cat:= [<i,j,Representative(poss(G,i,j))> : i,j in J | B[i,j] eq 1];
B:=Bm(G);
end while;
//Principle 2 applied to squares
for n in J do
for k in Sd(G,n) do
places:={<i,j> : i,j in J | Sn(i,j) eq n and  Matrix(Z,9,9,G)[i,j] eq 0
and k in Rd(G,i) meet Cd(G,j) };
if #places eq 1 then
Append(~G,<Representative(places)[1],Representative(places)[2],k>);
end if;
end for;
end for;
//Principle 2 applied to rows
for i in J do
for k in Rd(G,i) do
places:={<i,j> : j in J |  Matrix(Z,9,9,G)[i,j] eq 0 and
k in Sd(G,Sn(i,j)) meet Cd(G,j)};
if #places eq 1 then
Append(~G,<Representative(places)[1],Representative(places)[2],k>);
end if;
end for;
end for;
//Principle 2 applied to columns
for j in J do
for k in Cd(G,j) do
places:={<i,j> : i in J |  Matrix(Z,9,9,G)[i,j] eq 0 and
k in Sd(G,Sn(i,j)) meet Rd(G,i)};
if #places eq 1 then
Append(~G,<Representative(places)[1],Representative(places)[2],k>);
end if;
end for;
end for;
//check if any new entries have been filled out
B:=Bm(G);
d:=#G-g;
g:=#G;
until #G eq 81 or 0 in {B[i,j] : i,j in J} or d eq 0;
return G;
end function;

/* Procedure "su" tries to find a solution to sudoku problem G. It
first applies the function reduce. If this does not give a solution,
it goes through all entries in the matrix where more than one number
is possible.  It then writes each possibility in that entry, reduces
the new matrix, and checks if this is a solution.

If no solution is obtained in this way, one could iterate the process.
This version of the procedure, however, just stops here because this
seems to be sufficient in all known cases:
http://www.sudoku.org.uk/discus/messages/1/106.html?1144107305
*/
```

```
su:=procedure(G);
success:=Booleans()!false;
"The sudoku problem is", Matrix(Z,9,9,G);
G:=reduction(G);
if sol(G) then
"The easy solution is", Matrix(Z,9,9,G);
else
B:=Bm(G);
if exists{<i,j> : i,j in J | B[i,j] eq 0} then
"The sudoku problem has no solution";
else
if
exists(G1){reduction(Append(G,<i,j,k>)) : i,j,k in J |
           B[i,j] gt 1 and k in poss(G,i,j) and
           sol(reduction(Append(G,<i,j,k>)))} then
"The solution is", Matrix(Z,9,9,G1);
success:=true;
end if; end if;
if not success  then
print "This problem is too difficult for me";
end if; end if;
end procedure;


/* Example
G319:=[<1,3,7>, <1,4,5>, <1,6,6>, <1,9,3>,
       <2,1,4>, <2,5,3>, <2,7,2>,
       <3,2,8>,
       <4,2,2>, <4,4,7>, <4,5,9>, <4,7,5>, <4,9,4>,
       <6,1,5>, <6,3,4>, <6,5,1>, <6,6,2>, <6,8,7>,
       <7,8,8>,
       <8,3,5>, <8,5,7>, <8,9,9>,
       <9,1,6>, <9,4,4>, <9,6,1>, <9,7,7>];
time su(G319);
*/
```