/\*\*

數獨

\*/

package com.company;

import java.io.FileReader;

import java.io.\*;

import java.util.Scanner;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.lang.String;

public class Main {

public static void main(String[] args) {

String sudo\_str = read("sudoI");//數獨的檔案名稱

char[] sudo\_one\_arr = str\_to\_char(sudo\_str);

write("answer",""); // 檔案用來判定是否結束函式find\_answer

find\_answer(sudo\_one\_arr);

}

private static void find\_answer(char[] sudo\_one\_arr){

int[] empty\_arr = find\_empty(sudo\_one\_arr);

int sum\_of\_empty = empty\_arr.length;

char[][] possible\_two\_arr = new char[sum\_of\_empty][9];

int leastI = least\_possible\_number(sudo\_one\_arr);

String sudo\_str\_read = read("answer");

if(sudo\_str\_read.length() < 80) {

while (leastI == 1) {

int least\_locate = find\_the\_least\_locate(sudo\_one\_arr, leastI);

char[] list = judge(sudo\_one\_arr, least\_locate);

sudo\_one\_arr[least\_locate] = list[0];

leastI = least\_possible\_number(sudo\_one\_arr);

}

int emptyI = find\_empty(sudo\_one\_arr).length;

if (emptyI != 0) {

int least\_locate = find\_the\_least\_locate(sudo\_one\_arr,leastI);

char[] possible\_number = judge(sudo\_one\_arr,least\_locate);

char[][] possible\_answer = new char[leastI][81];

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

System.arraycopy(sudo\_one\_arr, 0, possible\_answer[i], 0, 81);

}

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

possible\_answer[i][least\_locate] = possible\_number[i]; }

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

int least = least\_possible\_number(possible\_answer[i]);

int empty = find\_empty(possible\_answer[i]).length;

if (least != 0 && empty != 0) {

find\_answer(possible\_answer[i]);

}

}

}

else if (judge\_answer(sudo\_one\_arr)) {

print\_the\_sudo(sudo\_one\_arr);

String sudo\_str = sudo\_str(sudo\_one\_arr);

write("answer", sudo\_str);}

}

}

private static int char\_find(char[] char\_list, char search){

for(int i=0; i<char\_list.length; i++){

if(char\_list[i] == search){

return i;}}

return -1;

}

private static boolean if\_content\_one\_to\_nine(char[] char\_list){

char[] number = {'1', '2', '3', '4', '5', '6', '7', '8', '9'};

for(char i:number){

if(char\_find(char\_list,i) == -1){return false;}

}

return true;

}

private static int[] find\_empty(char[] char\_list){

int empty\_sum = 0;

for(int i=0; i<char\_list.length; i++){

if(char\_list[i] == ' '){

empty\_sum++;}}

int[] empty\_arr = new int[empty\_sum];

for(int i=0,j=0; i<char\_list.length; i++){

if(char\_list[i] == ' '){

empty\_arr[j] = i;

j++;}}

return empty\_arr;

}

private static boolean judge\_answer(char[] sudo\_list){

for(int i=0; i<9; i++){

char[] number\_list = new char[9];

for(int j=0; j<9; j++){

number\_list[j] = sudo\_list[9\*i + j];}

if(!if\_content\_one\_to\_nine(number\_list)){return false;}}

for(int i=0; i<9; i++){

char[] number\_list = new char[9];

for(int j=0; j<9; j++){

number\_list[j] = sudo\_list[i + j\*9];}

if(!if\_content\_one\_to\_nine(number\_list)){return false;}}

int[] list = {0, 3, 6, 27, 30, 33, 54, 57, 60};

for(int i=0; i<9; i++){

char[] number\_list = new char[9];

for(int j=0; j<3; j++){

number\_list[j\*3] = sudo\_list[list[i] + j\*9];

number\_list[j\*3 +1] = sudo\_list[list[i] + j\*9 + 1];

number\_list[j\*3 +2] = sudo\_list[list[i] + j\*9 + 2];}

if(!if\_content\_one\_to\_nine(number\_list)){return false;}}

return true;

}

private static char[] str\_to\_char(String sudo\_str){

char[] sudo\_one\_arr\_temp = sudo\_str.toCharArray();

char[] sudo\_one\_arr = new char[81];

for(int i=0,j=0; i<sudo\_one\_arr\_temp.length; i++){

if(sudo\_one\_arr\_temp[i] != '\n'){

sudo\_one\_arr[j] = sudo\_one\_arr\_temp[i];

j++;}}

return sudo\_one\_arr;

}

private static String read(String file\_name){

StringBuilder content = new StringBuilder();

try{

FileReader file\_reader = new FileReader(file\_name + ".txt");

BufferedReader buffer\_reader = new BufferedReader(file\_reader);

String fr;

while((fr= buffer\_reader.readLine()) != null){

content.append(fr + '\n');

}

}

catch(IOException error){System.out.println(error);}

String contentI = content.toString();

return contentI;

}

private static void write(String file\_name, String content){

try{

FileWriter file\_writer = new FileWriter(file\_name + ".txt");

BufferedWriter buffer\_writer = new BufferedWriter(file\_writer);

buffer\_writer.write(content);

buffer\_writer.flush();

buffer\_writer.close();

}

catch(IOException error){System.out.println(error);}

}

private static char[] judge(char[] one\_arr, int locate){

StringBuilder horizontal\_gap = new StringBuilder();

int x = (locate - locate%9);

char[] horizontal = new char[9];

for(int i = x; i<(x+9); i++){

horizontal[i%9] = one\_arr[i];}

char[] number = {'1', '2', '3', '4', '5', '6', '7', '8', '9'};

for(char i:number){

if(char\_find(horizontal,i) == -1){

horizontal\_gap.append(i);}}

char[] horizontal\_gap\_arr = horizontal\_gap.toString().toCharArray();

StringBuilder vertical\_gap = new StringBuilder();

int y = locate%9;

char[] vertical = new char[9];

for(int i=0; i<9; i++ ){

vertical[i] = one\_arr[y + (i\*9)];}

for(char i:number){

if(char\_find(vertical,i) == -1){

vertical\_gap.append(i);}}

char[] vertical\_gap\_arr = vertical\_gap.toString().toCharArray();

int x\_nine, y\_nine;

x\_nine = ((x/9) - ((x/9)%3))/3;

y\_nine = (y - y%3)/3;

int loc = x\_nine\*27 + y\_nine\*3;

char[] nine\_nine = new char[9];

for(int i=0; i<3; i++){

nine\_nine[i\*3] = one\_arr[loc + 9\*i];

nine\_nine[i\*3 + 1] = one\_arr[loc + 1 + 9\*i];

nine\_nine[i\*3 + 2] = one\_arr[loc + 2 + 9\*i];

}

StringBuilder nine\_nine\_gap = new StringBuilder();

for(char i:number){

if(char\_find(nine\_nine, i) == -1){nine\_nine\_gap.append(i);}}

char[] nine\_nine\_gap\_arr = nine\_nine\_gap.toString().toCharArray();

StringBuilder total\_gap = new StringBuilder();

for(char i:number){

if(char\_find(horizontal\_gap\_arr, i) != -1 && char\_find(vertical\_gap\_arr, i) != -1 && char\_find(nine\_nine\_gap\_arr, i) != -1){

total\_gap.append(i);}}

char[] total\_gap\_arr = total\_gap.toString().toCharArray();

return total\_gap\_arr;

}

private static void print\_the\_sudo(char[] char\_list){

for(int i=0; i<81; i++){

System.out.print(char\_list[i]);

if(i%9 == 8){System.out.print("\n");}

}

}

private static int find\_the\_least\_locate(char[] char\_list, int least){

for(int i=0; i<81; i++){

if(char\_list[i] == ' '){

char[] possible\_list = judge(char\_list,i);

if(possible\_list.length == least){

return i;}}}

return -1;

}

private static int least\_possible\_number(char[] char\_list){

int least = 100;

for(int j=0; j<81; j++){

if(char\_list[j] == ' '){

char[] list = judge(char\_list,j);

if(list.length < least){least = list.length;}}}

return least;

}

private static String sudo\_str(char[] sudo\_one\_arr){

StringBuilder sudo\_str = new StringBuilder();

for(int i=0; i<81; i++){

sudo\_str.append(sudo\_one\_arr[i]);

if(i%9 == 8){sudo\_str.append('\n');}}

return sudo\_str.toString();

}

}