/\*

CSC 320 Project

Sudoku Solver

Sonia, Callum, Nelson, ChengXiang, Devroop

\*/

import java.util.ArrayList;

import java.util.Collections;

import java.util.Iterator;

public class SudokuSolver {

**/\* Method 1: Takes a string and return the same string if it’s “0” or decreases number on the string by 1 and returns it. - Sonia Barrios**

**\*/**

public static String minus1\_all(String triple){

if(triple.equals("0")){ **//Checks if the string is “0”**

return triple;

}

String conv\_triple = "";

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

conv\_triple= conv\_triple + Integer.toString(Character.getNumericValue(( triple.charAt(i))-1)); **// Decreases each character in the string by one.**

}

return conv\_triple;

}

**/\***

**Method 2 - Callum Thomas**

**Takes a string number in base 9 (Nonary) and converts it into a string in base 10 (decimal)**

**Either returns 0 or converts the string into decimal form and returns that string**

**Also pads the number with zeroes on the left.**

**\*/**

public static String nonal2dec\_triple(String triple){

if(triple.equals("0")){

return triple;

}

String nonal2dec = Integer.toString((Integer.parseInt(triple, 9)));

return String.format("%03d", Integer.parseInt(nonal2dec));

}

**/\***

**Method 3 - Devroop Banerjee**

**Takes string of number**

**Either returns 0 or adds a 1 (if it's a 3 digit number)**

**\*/**

public static String plus1(String triple){

if(triple.equals("0")){

return triple;

}

//return Integer.toString((Integer.parseInt(triple)+1));

String lala = String.format(“%03d”,(Integer.parseInt(triple)+1));

return lala;

}

/\* **Method 4 - Nelson Dai**

**Takes string of number**

**Either returns 0 or minus 1 (if it's a 3 digit number) \*/**

public static String minus1(String triple){

if(triple.equals("0")){

return triple;

}

String lalala = String.format(“%03d”,(Integer.parseInt(triple)-1));

return lalala;

}

**/\***

**\* Method 5 - Chengxiang Xiong**

**\* decimal number change to the number based 9**

\* \*/

public static int base9(int decimal){

if(decimal==0)

return 0;

int sign = 1;

if(decimal<0){

sign = -1;

decimal \*= -1;

}

ArrayList<String> degits = new ArrayList<String>();

while(decimal > 0 ){

degits.add(String.valueOf(decimal % 9));

decimal =decimal/9;

}

Collections.reverse(degits);

StringBuffer buf\_for\_degits = new StringBuffer();

Iterator<String> deg = degits.iterator();

boolean isNull = deg.hasNext();

while(isNull){

buf\_for\_degits.append(deg.next());

isNull = deg.hasNext();

}

String result = buf\_for\_degits.toString();

return Integer.parseInt(result) \*sign;

}

**/\* Method 6: Takes a string in base 10 and convert it to base 9.**

**Returns: the same string if it’s “0” or the nonary version of the input String. -Sonia Barrios \*/**

public static String dec2nonal\_triple(String triple){

if(triple.equals("0")){

return triple;

}

triple = Integer.toString(Integer.parseInt(triple, 10), 9); **// convert the string in base 10 to base 9.**

triple = String.format("%3s", triple).replace(" ", "0"); **// Replace with “0” if the string is less than 3 digits.**

return triple;

}

**/\***

**Method 7 - Callum Thomas**

**Adds one to each character in a triple value and returns it.**

**\*/**

public static String plus1\_all(String triple){  
 if(triple.equals("0")){  
 return triple;  
 }  
 String triple\_convert = "";  
 for(int i = 0; i < triple.length(); i++){  
 int trip = Character.getNumericValue(triple.charAt(i));  
 trip++; //Add one to the character converted to an integer.  
 triple\_convert = triple\_convert + trip;//Append the plus1ed characters  
 }  
 return triple\_convert;  
 }

**/\***

**Method 8 - Devroop Banerjee**

**Takes an array or list of strings and joins them by a space.**

**\*/**

public static String stringify(String[] splits){

String joined = String.join(" ", splits); **// joins the strings in the array**

return joined;

}

**/\* Method 9 - Nelson Dai**

**Applied dcmi notation negation to each variable in clause 123 414 222 0 => -123 -414 -222 0 \*/**

public int[] negate\_clause(clause){

String[] items = clause.split(" ");

Int[] array = new int[10];

for(int i =0; i<10; i++){

if (i != 0){

Array[i] = -(Integer.parseInt(clause[i]));

}else{

Array[i] = 0;

}

}

return array;

}

**/\* NEGATE\_CLAUSE string returning version by Callum Thomas**

**Simply adds a “-” to each clause. Returns the negated clause as a string**

**public static String negate\_clause(String clause){**

**String[] items = clause.split(" ");**

**String negated\_clause = "";**

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

**if(items[i].equals("0")){**

**negated\_clause = negated\_clause + "0"; //To make sure that the 0 isn’t -0**

**break;**

**}**

**negated\_clause = negated\_clause + "-" + items[i] + " ";**

**}**

**return nega**

**ted\_clause;**

**}**

**/\***

**/\***

**\* Method 10 - Chengxiang Xiong**

**\***

**\*/**

public static String dcmi\_format(String line) {

String[] splitLine = line.split("\\s+");

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

splitLine[i] = plus1(nonal2dec\_triple(minus1\_all(splitLine[i])));

}

return stringify(splitLine);

}

**/\* Method 11: Takes a String and separate the string by the space using an array. Return a String modified by minus1, dec2nona, plus1\_all. - Sonia Barrios**

**\*/**

public static String readable\_format(String line){

String[] splitLine = line.split("\\s+"); // creates an array with the strings

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

splitLine[i] = plus1\_all(dec2nonal\_triple(minus1(splitLine[i])));

}

return stringify(splitLine); // uses method 8 to join the strings by a space and return it.

}

**/\***

**Method 12 - Callum Thomas**

**Prints a header**

**/\***

public static void print\_header(){  
 System.out.println ("c cnf is file type, first num is variables, second num is clauses");  
 System.out.println ("p cnf 729 8577");

}

**/\***

**Method 13 - Devroop Banerjee**

**Reads file**

**Removes all symbols**

**Splits strings into substrings of 3**

**Replaces " " with a " 0"**

**\*/**

public static void print\_given\_cells\_requirement(){

**//Encoding for input cells**

**//Open a File**

try{

FileInputStream fstream = new FileInputStream(argv[0]);

DataInputStream in = new DataInputStream(fstream);

BufferedReader br = new BufferedReader(new InputStreamReader(in));

String line;

}catch (FileNotFoundException fileNotFoundException){

System.err.println("Error opening or creating the file");

System.exit(1);

}

String encoding = "";

i = 1;

j = 1;

line = line.replace("\n", "");

line = line.replace(" ", "");

line = line.replace("-", "");

line = line.replace("+", "");

line = line.replace("|", "");

while ((line = br.readLine()) != null){

for(j = 3; j < line.length; j+=4){

for(i = 0; i < line.length; i=j){

encoding += line.substring(i,j);

}

}

j += 1;

if (j > 9){

i += 1;

j = 1;

}

}

System.out.println("Given cells");

**//line is the values of the cells in readable format for a particular sudoku problem**

**//line = "146 164 179 197 225 237 253 323 341 392 412 438 469 491 521 587 616 645 674 693 711 765 782 851 876 889 915 934 947 962"**

converted\_line = dcmi\_format(line);

splits = converted\_line.replace(" ", " 0");

}

**/\* Method 14 - Nelson Dai**

**Prints the clauses for the requirement of each cell needing a value \*/**

**Had to make some debugging of your method nothing really big for testing method 16, here is the code: \*/**

**public static String negate\_clause(String clause){**

**String[] items = clause.split(" ");**

**int[] array = new int[10];**

**for(int i =0; i<10; i++){**

**if (i != 0){**

**array[i] = -(Integer.parseInt(items[i]));**

**}else{**

**array[i] = 0;**

**}**

**}**

**return ;**

**}**

**//Modified:**

**public static String negate\_clause(String clause){**

**String[] items = clause.split(" ");**

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

**if (!items[i].equals("0")){**

**items[i] = Integer.toString(-(Integer.parseInt(items[i])));**

**}**

**else {**

**items[i]="";**

**}**

**}**

**return stringify(items)+"0";**

**}**

public print\_cell\_requirement{

system.out.println(“c Output for first criteria. Each square needs a number.”);

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

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

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

clause= toDecimal(i, j, k);

out.write(String.valueOf(clause)+ " ");

}

out.write("0");

out.newLine();

}

}

}

**/\*Method 14 fixed\*/**

public static void print\_cell\_requirement() {

System.out.print("c Output for second criteria. Each row is unique.");

System.out.println();

ArrayList<String> clause = new ArrayList<String>();

ArrayList<String> term = new ArrayList<String>();

for (int i = 1; i < 10; i++) {

for (int j = 1; j < 10; j++) {

clause.clear();

for (int k = 1; k < 9; k++) {

clause.add(String.format("%s%s%s", i, j, k));

clause.add("0");

term.add(stringify(clause.toArray(new String[0])));

}

}

}

for (int i = 0; i < term.size(); i++) {

String st = dcmi\_format(term.get(i));

System.out.println(st);

}

}

**/\***

**\* Method 15 - Chengxiang Xiong**

**\* \*/**

public static void print\_row\_requirement() {

System.out.println();

System.out.print("c Output for second criteria. Each row is unique.");

ArrayList<String> clause = new ArrayList<String>();

ArrayList<String> term = new ArrayList<String>();

for (int i = 1; i < 10; i++) {

for (int j = 1; j < 10; j++) {

for (int k = 1; k < 9; k++) {

for (int l = i + 1; l < 10; l++) {

clause.add(String.format("%s%s%s", k, i, j));

clause.add(String.format("%s%s%s", l, i, j));

clause.add("0");

term.add(stringify(clause.toArray(new String[clause.size()])));

}

}

}

}

for (int i = 0; i < term.size(); i++) {

System.out.print(negate\_clause(term.get(i)));

}

}

**/\* Method 16: Doesn’t take any parameters -Sonia Barrios**

**///Rough method (need testing and debugging)**

**\*/**

public static void print\_col\_requirement() {

System.out.println("c Output for the third criteria. Each column is unique");

ArrayList<String> clause = new ArrayList<String>();

ArrayList<String> term = new ArrayList<String>();

for(int i=1; i<10; i++){

for(int j=1;j<10; j++){

for(int k=1; k<9; k++){

for( int l=i+1; l<10; l++){

clause.add(String.format("%s %s %s", k, i, j));

clause.add(String.format("%s %s %s", l, i, j));

clause.add("0");

term.add(stringify(clause.toArray(new String[clause.size()])));

clause.clear();

}

}

}

}

for(int i=0; i< term.size(); i++){

System.out.println(negate\_clause(dcmi\_format(term.get(i))));

}

}

**/\*Method 17 - Callum Thomas**

**Prints the subgrid 3x3 uniqueness requirement clauses.**

**\*/**

public static void print\_subgrid\_requirement() {

System.out.println();

System.out.println("c Output for the fourth criteria. Each 3x3 cell is unique");

ArrayList<String> term = new ArrayList<String>();

for(int k=1; k<10; k++){

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

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

for(int u=1; u<3; u++){

for(int v=1; v<4; v++){

for(int w=u+1; w<4; w++){

for(int t=1; t<4; t++){

if(v != t){

ArrayList<String> clause = new ArrayList<String>();

clause.add(String.format("%s %s %s", 3\*a+u, 3\*b+v, k));

clause.add(String.format("%s %s %s", 3\*a+w, 3\*b+t, k));

clause.add("0");

term.add(stringify(clause.toArray(new String [clause.size()])));

}

}

}

}

}

}

}

}

for(int i=0; i< term.size(); i++){

System.out.println(negate\_clause(term.get(i))); // do you have to converted to cdmi\_format?

}

}

**/\***

**Method 18 - Devroop Banerjee**

**Main.... Self explanatory**

**\*/**

public static void main(String[]args){

print\_header();

print\_given\_cells\_requirement();

print\_cell\_requirement();

print\_row\_requirement();

print\_col\_requirement();

print\_subgrid\_requirement();

}

}