Problem Set 4: 数独

due by 7:00 P.M. on Friday, 24 October 2008

Goals.

- Learn to use ncurses, a library for GUIs.
- Design and implement larger pieces of software.
- Master sudoku.
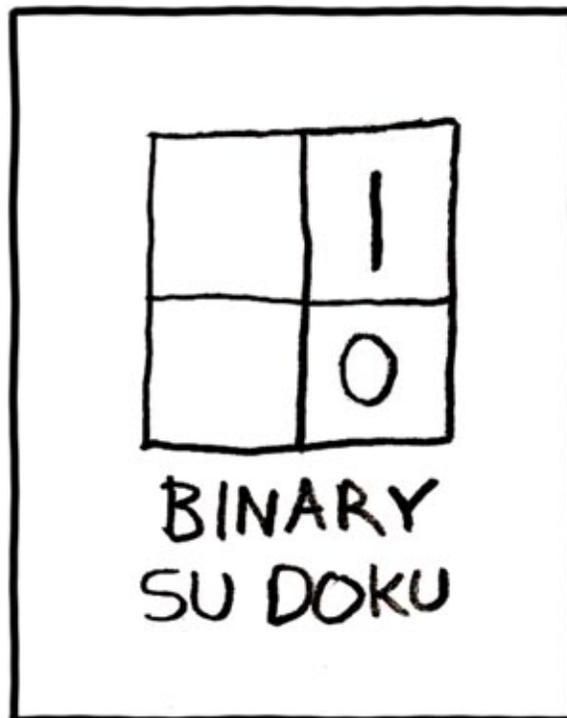
Recommended Reading.

- Sections 1 − 13 of `http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/`.

image from `http://xkcd.com/74/`

Academic Honesty.

All work that you do toward fulfillment of this course's expectations must be your own unless collaboration is explicitly allowed (*e.g.*, by some problem set or the final project). Viewing or copying another individual's work (even if left by a printer, stored in an executable directory, or accidentally shared in the course's virtual terminal room) or lifting material from a book, magazine, website, or other source—even in part—and presenting it as your own constitutes academic dishonesty, as does showing or giving your work, even in part, to another student.

Similarly is dual submission academic dishonesty: you may not submit the same or similar work to this course that you have submitted or will submit to another. Moreover, submission of any work that you intend to use outside of the course (*e.g.*, for a job) must be approved by the staff.

You are welcome to discuss the course's material with others in order to better understand it. You may even discuss problem sets with classmates, but you may not share code. In other words, you may communicate with classmates in English, but you may not communicate in, say, C. If in doubt as to the appropriateness of some discussion, contact the staff.

You may even turn to the Web for instruction beyond the course's lectures and sections, for references, and for solutions to technical difficulties, but not for outright solutions to problems on problem sets or your own final project. However, failure to cite (as with comments) the origin of any code or technique that you do discover outside of the course's lectures and sections (even while respecting these constraints) and then integrate into your own work may be considered academic dishonesty.

All forms of academic dishonesty are dealt with harshly.

Grades.

Your work on this problem set will be evaluated along three primary axes.

Correctness. To what extent is your code consistent with our specifications and free of bugs?
Design. To what extent is your code written well (*i.e.*, clearly, efficiently, elegantly, and/or logically)?
Style. To what extent is your code readable (*i.e.*, commented and indented with variables aptly named)?

Getting Started (in teh Cloud).

0. For this problem set, we're going to have you use `cloud.cs50.net`, our new cluster "in the cloud," instead of `nice.fas.harvard.edu`.

    Thanks to Amazon's Elastic Compute Cloud (EC2), we're in the process of replacing the physical servers on which the course has relied for years (*i.e.*, `nice.fas.harvard.edu`) with "virtual machines" owned by Amazon but controlled by us.

    If unfamiliar with virtual machines, head on over to Wikipedia for a few minutes:

    `http://en.wikipedia.org/wiki/Virtual_machine`

    And also read up on cloud computing itself:

    `http://en.wikipedia.org/wiki/Cloud_computing`

    Then read up on the specific cloud that you're about to be in:

    `http://aws.amazon.com/ec2/`

    Truth be told, we're quite excited by what we'll be able to do this semester with all this technology. Not only will EC2 allow the course to grow and shrink its infrastructure as problem sets demand, it will also enable us, toward term's end, to provide some students with virtual machines of their own on which they have "root" (*i.e.*, administrator) access for their final projects. All this for just $0.10 per hour. Not bad!

    Much like you have an account on `nice.fas.harvard.edu`, you're about to have an account on `cloud.cs50.net` too. For simplicity, your username on `cloud.cs50.net` will be identical to your FAS username, but realize that they are different accounts (with different home directories). If you change your password on one, it will not change on the other. If you edit some file on one, it will not change on the other. Functionally, though, you should find `cloud.cs50.net` to be quite similar to `nice.fas.harvard.edu`.

    Even though we've been beta-testing the cluster on those who tackled the Hacker Editions of Problem Sets 2 and 3, we do hope you'll forgive if any technical difficulties sneak up on us this week! Glenn, Keito, and I (the course's sysadmins) will be watching the course's bulletin board constantly, as we're eager for feedback, particularly if things break.

If curious, here's a sketch of what you're about to try out!

1. Head on over to

   `http://cs50.net/me/`
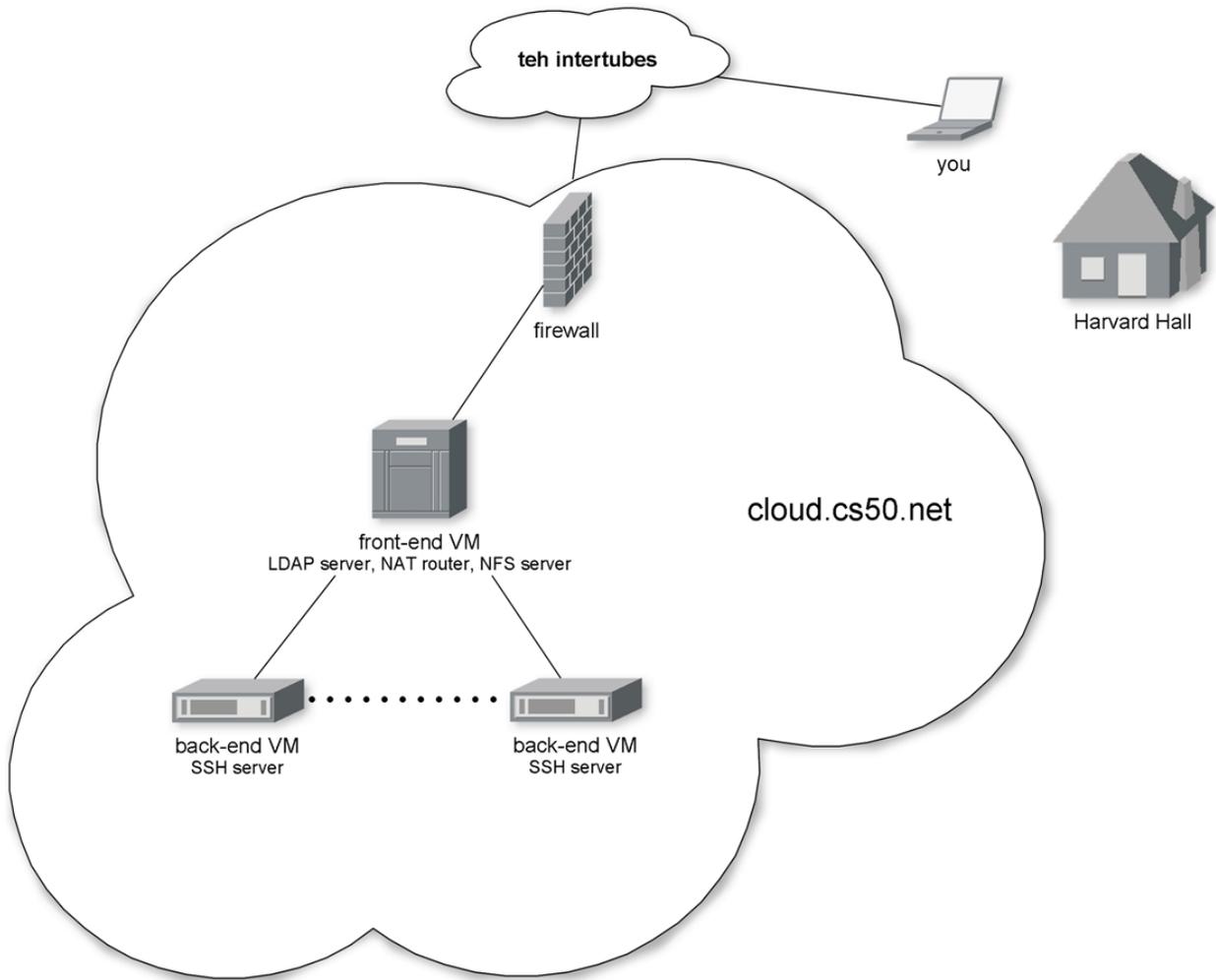
   to find out your password on `cloud.cs50.net`. Lest you run into a chicken-and-egg problem, you'll want to log into the course's website using your FAS username and password when prompted. Once you know your initial password on `cloud.cs50.net`, you can start logging into the course's website with it or with your FAS password.

2. Now go ahead and actually SSH to `cloud.cs50.net` (much like you would to `nice.fas.harvard.edu`), create a directory called `pset4` in your brand-new home directory, and then navigate your way to `~/pset4/`. (Remember how?) All of the work that you do for this problem set should reside in this directory so that we know where to look if we need lend you a

hand. No need to put `pset4/` in a `~/cs50/` subdirectory like you did on `nice.fas.harvard.edu`. We know it's for 50!

Your prompt should now resemble the below.

```
username@cs50.net (~/pset4):
```

If you'd like to change your password, execute

```
passwd
```

at your prompt. When prompted for your login(LDAP) password, provide your current (*i.e.*, initial) password for the cloud. You'll then be prompted up to four times for your new password: enter your new password exactly the same all four times.

Now go ahead and execute the command below.

```
cp ~cs50/pub/distributions/pset4/* ~/pset4/
```

That command will copy the distribution code in the staff's `pset4/` directory into your own `~/pset4/` directory. Navigate your way to your copy by executing the command below.

```
cd ~/pset4/
```

If you list the contents of your current working directory, you should see the below. If you don't, don't hesitate to ask the staff for assistance.

```
l33t.bin  Makefile  n00b.bin  sudoku.c  sudoku.h
```

Well those look like fun!

store.cs50.net.

3. So, believe it or not, we've decided to get on that Internet and set up a shop.[1] Debuting soon at http://store.cs50.net/ will be CS 50 merchandise, including t-shirts and more, thanks to CafePress.com.[2] But we need help with designs.[3] Best to keep your design's focus on 50 and not on Harvard itself, as there are a few policies:

```
http://provost.harvard.edu/policies_guidelines/names_insignias.php
```

---

[1] That's so us, isn't it.

[2] Who wouldn't want to wear a t-shirt that says **I SURVIVED CS50** or, better yet, **PWNED BY CS 50**?

[3] Aaron is all out of lolcansus.

CafePress.com will let us put JPEGs on a whole bunch of merchandise, provided your design adheres to the requirements here:

`http://www.cafepress.com/cp/info/sell/images/images_onesize`

So! If you would like to submit a possible design for some merchandise, email a 2000px by 2000px JPEG (*i.e.*, 10" by 10" at 200 DPI) to **store@cs50.net** by this problem set's deadline!

The numbers must be single.

4.   You're about to get addicted to homework.

5.   Much like I catch on to Internet memes two years after they're funny, similarly did I catch on to Sudoku about two years after it became hugely popular in this country, at which point I lost a week of my life playing it constantly on my BlackBerry. Your turn!

Much like the Game of Fifteen, Sudoku is a game of logic involving numbers. But it's a great deal more fun. Consider the puzzle below.

1			3	4				5
					6			8
				5			6	3
		1	6		5	8		9
		3		7		5		
9		6	2		1	3		
6	8			2				
3			5					
2				1	9			7

The object of Sudoku is to fill this 9×9 grid in such a way that each column, each row, and each of the nine 3×3 boxes therein contain each of the numbers 1 through 9 exactly once. A whole bunch of strategies exist, but the general idea is to figure out iteratively what numbers could go where.

For instance, let's home in on one of the 3×3 boxes that already has a lot of numbers and work the ol' process of elimination. How about the box in the middle, highlighted below.

1			3	4				5
					6			8
				5			6	3
		1	6		5	8		9
		3		7		5		
9		6	2		1	3		
6	8			2				
3			5					
2				1	9			7

Let's see, that box already has a 1, a 2, but no 3. Where could we put 3? Well, 3 can't go in that box's bottom row, since the box to the right already has a 3 in that row. And 3 can't go on either side of the 7, since the box to the left already has a 3 in that row. Aha! It must be that 3 belongs in that box's top row, in which case there's only one place to put it! And so we fill in that spot with a 3, per the below.

1			3	4				5
					6			8
				5			6	3
		1	6	3	5	8		9
		3		7		5		
9		6	2		1	3		
6	8			2				
3			5					
2				1	9			7

Let's try another trick now. Rather than figure out where a number can go, let's figure out where a number cannot! Let's home in on 9. Highlighted in gray now are all of the spots that 9 cannot go, either because there's already another number there or because there's already a 9 in the highlighted row, column, or box, per the below.

1			3	4				5
					6			8
				5			6	3
		1	6	3	5	8		9
		3		7		5		
9		6	2		1	3		
6	8			2				
3			5					
2				1	9			7

Well, look at that! Looks like we've found a home for 9 within that box in the middle because there's only one place it can possibly go, per the below.

1			3	4				5
					6			8
				5			6	3
		1	6	3	5	8		9
		3	9	7		5		
9		6	2		1	3		
6	8			2				
3			5					
2				1	9			7

Rinse, wash, and repeat these sorts of tricks enough times, and (assuming no PEBPAC) we'll end up with the solution below.[4]

1	6	8	3	4	2	7	9	5
5	3	2	7	9	6	1	4	8
7	9	4	1	5	8	2	6	3
4	7	1	6	3	5	8	2	9
8	2	3	9	7	4	5	1	6
9	5	6	2	8	1	3	7	4
6	8	7	4	2	3	9	5	1
3	1	9	5	6	7	4	8	2
2	4	5	8	1	9	6	3	7

If still not quite clear on how the game is played, feel free to turn to the Wikipedia.

```
http://en.wikipedia.org/wiki/Sudoku
```

And if interested for your own edification in the mathematics and algorithmics behind the game, you might also find these articles of interest:

```
http://en.wikipedia.org/wiki/Mathematics_of_Sudoku
http://en.wikipedia.org/wiki/Algorithmics_of_sudoku
```

6. Just for fun, we've included a whole bunch of puzzles (and solutions) at this document's end. And if you start finding yourself particularly addicted, odds are an implementation of sudoku exists for your mobile phone!

7. Alright, now the real fun begins. You're about to implement (most of) Sudoku in C.

Much like we provided you with some code for the Game of Fifteen, similarly have we provided you with a skeleton for Sudoku. Whereas Problem Set 3 relied on ANSI escape sequences to implement the Game of Fifteen's "graphics," though, this problem set introduces a library called "ncurses" (formerly called "curses") that will provide your implementation of Sudoku with a nicer GUI (graphical user interface). To be sure, your program won't look like Mac OS or Windows, but it should look sexier than that Game of Fifteen![5] Not only does ncurses make it pretty easy to integrate colors into a program (and even dialogs and menus), it also allows you to treat your terminal window as a grid of `chars`, any one of which can be updated without affecting the others. That sort of feature is perfect for a game like Sudoku, as you'll be able to add numbers to the game's board one at a time without having to re-generate the whole screen after each move (as you did with `printf` for the Game of Fifteen).

---

[4] The second 'P' stands for "puzzle," and you're in the chair.
[5] Shuttleboy, for instance, uses ncurses. And he sure looks good.

Now, a typical terminal window is 80 characters wide by 24 characters tall (*i.e.*, 80×24), and ncurses addresses those characters by way of (y, x) coordinates, whereby (0, 0) refers to your window's top-left corner, (0, 79) refers to your window's top-right corner, (23, 0) refers to your window's bottom-left corner, and (23, 79) refers to your window's bottom-right corner.[6] Even if your window boasts dimensions smaller or larger than these, the idea is the same. When it comes time to fill in a blank with respect to Sudoku, you'll simply update the `char` at some (y, x) coordinate.

Now, how about that skeleton. Essentially, we've implemented an aesthetic framework for the game so that you can focus on the more interesting parts: namely, the game's features. In fact, we've written the code (and comments) in such a way that you should be able to learn quite a bit about ncurses and more simply by reading our code. And you'll find that we've structured Sudoku's framework much like we did the Game of Fifteen's. It's in `main` that we have a big loop, waiting and waiting for some user's input. And it's in separate functions that we (and, soon, you) set the game up and respond to that input.

Because this game is meant to be more sophisticated (and fun) than the last, you'll also find that we've given you more code this time. Don't freak out, but it's just over 600 lines. But know now that none of it is all that complicated. In fact, if you look at each of the functions in isolation, you'll likely find each pretty straightforward. What's neat is that when you combine so many building blocks, you get some pretty compelling results. In fact, let's take a look.

If you're not still there, navigate your way to `~/pset4/` on `cloud.cs50.net` and execute the (increasingly familiar) command below.

```
make
```

You should find a brand-new executable called `sudoku` in your current working directory. Go ahead and run it by typing the command below.

```
sudoku
```

You won't yet see our skeleton but instead the game's usage:

```
Usage: sudoku n00b|l33t [#]
```

Not only does our skeleton support two levels of game play (n00b and l33t), it also comes with 1024 different boards for each level. Ultimately, if you'd like to play a pseudorandomly chosen n00b board, you'll want to execute just:

```
sudoku n00b
```

---

[6] Annoyingly, yes, it's (y, x) and not (x, y).

But if you want to play a specific board (*e.g.*, n00b #42), perhaps one that defeated you earlier, you can load it up manually:

```
sudoku n00b 42
```

Alright, go ahead and execute

```
sudoku n00b 42
```

to fire up our skeleton with n00b #42. You should see a GUI like that below.[7]
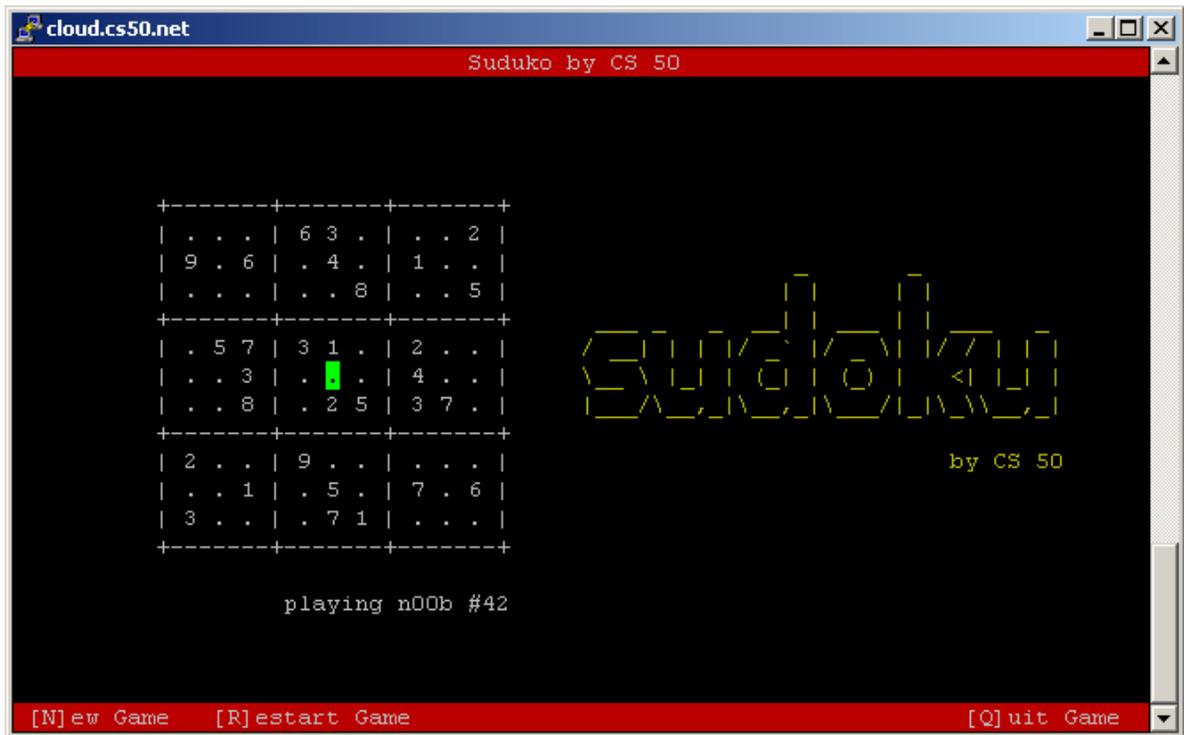
Notice how, for clarity's sake, we use periods for blanks; underneath the hood, we represent each of those same blanks with 0 (an actual `int`). So this is all pretty neat, but this skeleton lacks that personal touch (not to mention support for moving the cursor). What do work out of the box are **[N]ew Game**, **[R]estart Game**, and **[Q]uit Game**. So go ahead and hit **Q** to quit.

Then open up `sudoku.h` (as with Nano). You'll find in that file a whole bunch of constants that get compiled into the program. Go ahead and change, at least, `AUTHOR` to your own name. Feel free to change `TITLE` as well. To see the results, save your changes and quit. Then re-run `make` followed by `sudoku`. You've just made the program your own!

---

[7] If using PuTTY on Windows or Terminal on Mac OS, you should see a colorful GUI. If you're instead using SecureCRT on Windows and only see monochrome (*e.g.*, black and white) or gaps in the program's borders, go to **Options** → **Session Options...** → **Terminal** → **Emulation**, and change **Terminal** to **Linux** and check the box next to **ANSI Color**. After clicking **OK**, go to **Options** → **Save Settings Now**. You may then need to log out and back in to see our GUI in color. If you can't get color to work, head to the course's bulletin board for advice!

Now go back into `sudoku.h` and play with all those mentions of color. It turns out that ncurses deals with colors in pairs, whereby characters have both a foreground color and a background color. By default, characters' foregrounds are white and backgrounds are black. But clearly we've overridden those defaults for our skeleton's borders and logo. For now, you'll want to leave that `enum` alone, but feel free to change the values of any constants whose names begin with `FG_` or `BG_`. Here are the colors that ncurses comes with:

```
COLOR_BLACK
COLOR_RED
COLOR_GREEN
COLOR_YELLOW
COLOR_BLUE
COLOR_MAGENTA
COLOR_CYAN
COLOR_WHITE
```

You will, of course, need to recompile your game to see any colorful changes. Not all that hard to make one hideous game, eh?

Okay, now take a quick glance at `Makefile`. It should look pretty similar to the Game of Fifteen's, but take note that we've included a `-lncurses` flag. And be sure not to change the two tabs in that file to spaces!

Now take a look at, say, `n00b.bin`, but not with Nano this time! Instead, execute the command below.[8]

```
xxd -b n00b.bin
```

Wow, a whole lot of numbers probably flew past. You've just looked at the contents of a binary file. Inside that file are a whole bunch of 32-bit `ints`, $1024 \times 81 = 82,944$ of them, in fact, as that file contains 1024 n00b boards, each of which includes 81 numbers and/or blanks (for a 9×9 grid).[9] Similarly does `l33t.bin` contain 1024 l33t boards.

Now that you've run `sudoku` at least once, you might also have noticed a file called `log.txt` that wasn't there when you first copied our code over. You're welcome to look, but you needn't pay much attention; it's generated by our framework in order to facilitate automated tests of your code.

Alright, we're really moving along. Only one file to go!

---

[8] Note that "n00b" is spelled with two zeroes!

[9] We could have used `unsigned chars` instead of `ints`, since Sudoku only needs numbers from 1 to 9, but we decided that `ints` would be simpler, despite the additional cost in space.

8.   Damn, it's the big one.

Go ahead and open up `sudoku.c` (as with Nano).

The best way to tackle this problem set is to start by understanding this file. We'll get you started.

First take note of one of the file's first lines:

```
#define CTRL(x) ((x) & ~0140)
```

Just as you can define what we know as constants with `#define`, you can also define "macros," short snippets of code that behave a little bit like functions but without the overhead (*e.g.*, stack frames) of an actual function call. This particular macro will enable you to detect control characters from users, if you so desire. Out of the box, our skeleton already understands ctrl-L, a keystroke meant to induce a redrawing of the game's screen.

Now take a look at the `struct` just below that macro. Think of a `struct` as a wrapper that groups related variables together. Inside this particular `struct` is a whole bunch of globals. Truth be told, we could have defined all those variables as globals without using a `struct`, much like we did for the Game of Fifteen. But because there are so many, all related to this game, we decided to "keep them together" in one big struct called `g`. That way, it'll be all the more obvious when these variables are used that they're not, in fact, locals.

Next notice our skeleton's prototypes. Much like we divided the Game of Fifteen into functions whose names described their particular role, similarly have we taken that approach here. But more on those later.

Now dive into `main`. Best, though, if we not hold your hand too much through this one. We daresay that learning to program is as much about writing your own programs as it is about reading others', particular when your assignment (or job) is to build on the latter. Odds are you'll thank us some day for actually having comments in ours![10]

To be sure, there's a lot going on in this file, but you don't need to read each and every line (yet) to get a sense of the program's overall flow.

Do read each and every line in `main`, though. After all, that's the function that drives this whole program. And because our other functions' names rather say what those functions do, you can probably read `main` from top to bottom and have a pretty good idea of how the program currently works.

Next dive into some of the functions that `main` calls. A good one to start with is `startup`, as it gets ncurses going. Notice how it calls a bunch of other functions that appear to configure ncurses. Although we've commented each call, you might want to pull up the man page for some or all of those functions, if only to get all the more comfortable with ncurses.

---

[10] You should see the mess that is the open-source bulletin board software we're using.

Next take a look at `load_board`, the function that loads a n00b or l33t board from disk, depending on the value, if any, in `main`'s `argv[2]`. You needn't understand how `fopen`, `fseek`, `fread`, or `fclose` work for this problem set, but it is kind of neat how they get all those bits into memory. What this function ultimately does is load 81 `ints` into the global array called `g.board`. Not all that hard!

Let's see, next take a look at `draw_borders`. It's this function that creates the game's, well, borders. Of particular note in this function is how to use ncurses. Notice, for instance, that the function first determines your terminal window's dimensions using a macro called `getmaxyx` (that comes with ncurses).[11] It eventually uses those maxima to fill your window's topmost and bottommost rows with some color (and instructions). Notice how the function enables color, specifically turning on the `COLOR_PAIR` attribute that we called `PAIR_BORDER` (back in `suduko.h`). It then proceeds to draw the game's borders by moving, left to right, from coordinate to coordinate, laying down blank spaces. (Because we, at least, defined `PAIR_BORDER` with a red background, the results are "blank" lines filled with color.) But the function next lays down some text, centering your program's `TITLE` and `AUTHOR` in the topmost border using some simple arithmetic. It then plants some instructions on top of bottommost border, before turning color back off.

Now take a look at `draw_grid`. It's this function that lays down the ASCII art that represents our game's board. Similarly does it first determine your window's dimensions. It then uses those values to determine coordinates for the grid's top-left corner. (We decided that we wanted the grid roughly in the middle of your window but slightly to the left, and so we came up with those formulas by trial and error.) Rather than generate this grid character by character, this function instead lays down whole strings instead (using ncurses' `mvaddstr` function). Specifically, this function moves the cursor to a specific coordinate and adds a string there. It then does that again and again (in that `for` loop) in order to print most of the game's grid. (Again, we determined most of those coordinates by trial and error.) We then thought it'd be neat to remind the user of the level and board that he or she is playing, and so we constructed a string on the fly using `sprintf`, and then added it to the screen with a final call to `mvaddstr`.

Incidentally, if curious to learn more about all these ncurses functions, `man` is your friend.[12]

Next glance at `draw_logo`. Notice how it bases its own coordinates on those of the grid. Notice, too, how distorted our logo looks. That's because we had to escape some of its backslashes with backslashes of our own! Do feel free to alter the logo. You might find this site a fun time:

`http://www.network-science.de/ascii/`

Now look at `draw_numbers`. It's this function that fills that otherwise empty grid with the numbers in `g.board`. That kind of knowledge is bound to be useful!

---

[11] It's because `getmaxyx` is a macro and not an actual function that you don't need to pass in its arguments by reference.
[12] Some functions don't actually have their own man page, so you might need to execute `man ncurses`, and then look for the **Manual Page Name** for some the **curses Routine Name** of interest.

Now glance at `show_banner` and `hide_banner`. Both pretty simple, these functions exist so that you can show (and hide) messages to users. In fact, while using ncurses, do not use `printf` as well. Bad things will happen.

Speaking of `show_banner`, why don't we also peek at `show_cursor`. Recall that functions like `mvaddch` and `mvaddstr` end up moving your cursor in order to add text to the screen. That's kind of annoying if you want to use that same cursor to play the actual game. And so it's necessary to remember where the cursor should be with respect to that grid. Glance back at that global called `g` and you'll see how we do it. This `show_cursor` function relies on that `struct` to return the cursor to where it should be after screen updates.

Let's see, you needn't worry too much about `handle_signal`. Just know that when terminal windows are resized (as from 80×24 to something larger or smaller), "signals" are generated. Our code is "listening" for that signal so that we can respond to resizings by re-centering everything.

Okay, so that only leaves `log_move`, `redraw_all`, `restart_game`, and `shutdown`. Those you can handle! Take the same approach that we took here, walking through each function, pulling up man pages as needed, and only move on once you understand each function's flow.

So that's everything. Not bad for 600+ lines.

9. And yet none of those lines actually implement Sudoku. But that's where you come in! Your challenge for this problem set is to implement a few features, among them support for actual game play! Specifically, you must implement each of the REQUIRED FEATURES below and any one (1) of the ADDITIONAL.

   This problem set is perhaps more about design than anything else, so do give some thought about how best to implement some feature, given the game's framework. With that said, you are welcome to change most any aspect of our code if the change fits your design better. However, what you must not change is anything related to logging, including `log_move`. So that we can automate some tests of your code, your program, no matter your changes, must still call our implementation of `log_move` after each keystroke from users, whether or not that keystroke actually altered the board.

   Alright, get to it! Here's your menu of features. Know that we've enumerated the REQUIRED in the order in which they should probably be implemented.

   REQUIRED FEATURES

   ❑ At the moment, the cursor is "stuck" in the board's center. Enable users to move that cursor up, down, left, and right by way of their keyboard's arrow keys. You're welcome to support other keys for movement as well, but you must support `KEY_UP`, `KEY_DOWN`, `KEY_LEFT`, and `KEY_RIGHT`, constants that represent the characters fed to ncurses' `getch` function when arrow keys are pressed. (See `getch`'s man page for even more constants.) Only allow the user to move his or her cursor to coordinates where there are actual numbers or blanks. (For the sake of discussion, let's call those 81 locations "cells.") The

cursor should "hop over" one-character lateral gaps between cells as well as the innermost crossbars that make up the grid's lines. Even though you might be tempted to hop over numbers that came with the board (*i.e.*, that cannot be changed), resist the temptation; allow the cursor to be in any one of those 81 cells.

❑ Enable the user to replace any blank with a number by moving his or her cursor over that blank and then hitting a number from 1 to 9.

❑ Enable the user to change a number that he or she already inputted back to a blank (by hitting 0, a period, Backspace, or Delete) or to some other number from 1 to 9. But do not allow the user to alter numbers that "came with" the board.

❑ Any time the user changes the board, check whether the game has been won. If so, display a congratulatory banner and prevent the user from changing the board further.

❑ Any time the user changes the board, check whether he or she has inserted a number where it cannot possibly go (because that same number already exists in the same column or row or 3×3 box). If so, display a banner that warns of the gaffe that disappears after the user's next move (assuming the next move is not also an obvious mistake).

IMPLEMENT AT LEAST ONE (1) OF THESE ADDITIONAL FEATURES
And make clear in the comments atop `sudoku.c` which of these features you implemented.

❑ In addition to displaying a congratulatory banner, turn all 81 numbers green when the game has been won.

❑ In addition to warning the user of obvious mistakes with a banner, turn the column, row, or 3×3 box with the error red until the mistake is corrected.

❑ Enable the cursor to "wrap around" from the top row to bottom, bottom to top, left to right, or right to left if the user presses `KEY_UP`, `KEY_DOWN`, `KEY_LEFT`, or `KEY_RIGHT`, respectively, one too many times.

❑ Display numbers that "came with" the board in a different color than those that the user has inputted.

❑ Keep track (in seconds) of the amount of time that the user has been playing the current board and allow the user to show or hide that clock at any time by hitting T. Be sure to stop the clock the moment the game has been won.

❑ Allow the user to undo the last change made to the board by hitting U or ctrl-Z.

10. Don't forget about the course's **bulletin board**! And don't forget to check in your code with **ci**!

     If you'd like to play with the staff's own implementation of `sudoku` on `cloud.cs50.net`, you may execute the below.

     `~cs50/pub/solutions/pset4/sudoku`

Submitting Your Work.

11. Ensure that your work is in `~/pset4/` on `cloud.cs50.net`. Submit your work by executing the command below.

     `cs50submit pset4`

     Thereafter, follow any on-screen instructions until you receive visual confirmation of your work's successful submission. You will also receive a "receipt" via email to your FAS account, which you should retain until term's end. You may re-submit as many times as you'd like; each resubmission will overwrite any previous submission. But take care not to re-submit after the problem set's deadline, as only your latest submission's timestamp is retained.

Puzzle 1 (Medium, difficulty rating 0.55)

	4		7				9	
							4	7
7				4	6	5		
2			8			9	3	1
		8				4		
1	7	4			3			2
		7	4	3				6
8	9							
	1				2		5	

Puzzle 2 (Easy, difficulty rating 0.39)

5	3					7		8	
5	3				7		8	
			8			9		4
1				4	3	7		
					9	1		
9	7						2	6
		3	1					
		6	3	1				2
2		9		4				
	1		6				4	7

Puzzle 3 (Hard, difficulty rating 0.75)

9			8	7		1		6
	6				1	8		
		8						
5				1	9			
	8	4	7		5	2	1	
			3	2				4
					4			
		6	5				7	
7		5		4	3			8

Puzzle 4 (Medium, difficulty rating 0.49)

3	5	8	9	6		2	4	
7		9	4					
			3		7			
				7				1
		4				6		
8				3				
			5		6			
					9	4		2
	1	5		2	3	8	9	6

Puzzle 5 (Hard, difficulty rating 0.61)

					4		5	
		5		6	9			
2	3		8	5		9		6
9					6			2
		7				5		
5			4					7
1		9		8	2		7	5
			6	9		2		
	6		7					

Puzzle 6 (Medium, difficulty rating 0.53)

	9							
		3				4	1	9
		1	9	5	4	2		
9				8			2	
5			4	2	9			6
	2			1				5
		5	7	6	3	1		
1	3	9				5		
							8	

Puzzle 7 (Medium, difficulty rating 0.47)

	1			5			2	
		5	6			1	8	
	6	7			9			5
1			7				6	
	3						7	
	9				1			8
6			8			2	1	
	2	9			3	6		
	7			2			9	

Puzzle 8 (Hard, difficulty rating 0.71)

4			8			6		
				5				4
	8		6		4		1	
		8	3	1		2		
3		2				8		1
		6		4	8	7		
	1		4		6		9	
2				9				
		5			1			3

Puzzle 9 (Easy, difficulty rating 0.35)

5				2			3	
	8							
7	6		1	3		4		5
6				1	5		2	
	1						5	
	4		8	6				3
9		1		4	3		8	2
							6	
	3			7				9

Puzzle 10 (Easy, difficulty rating 0.43)

6		4	9		7		8	
9	8							3
					3			4
		5		3	9	8		
	9			7			3	
		8	5	6		7		
4			3					
2							4	8
	6		2		5	9		7

Puzzle 11 (Medium, difficulty rating 0.56)

6	2		3			5		7
			1				2	3
				5		6		4
1					9			5
		6		3		4		
4			5					8
2		9		1				
3	6				7			
8		1			5		7	9

Puzzle 12 (Easy, difficulty rating 0.35)

5			7		9			
7					8	3		4
9			4				6	
	5	8	1	4				
		7				4		
			9	7	5	1		
	3			6				2
8		5	9					7
			5		3			6

Puzzle 13 (Medium, difficulty rating 0.50)

3	1				5	6		
				6	8	1		
8						7		
			5		3	4		8
1		9				2		6
4		8	6		9			
		7						1
		1	8	4				
		3	7				6	5

Puzzle 14 (Medium, difficulty rating 0.53)

5							8	2
		8		9	6			7
	1		8	3				
				8	9			
6	4		2	7	1		3	8
			6	5				
			1	3			4	
1			9	6		7		
7	5							9

Puzzle 15 (Easy, difficulty rating 0.37)

			4				1	5
	9	5	6	2				
		2	8					9
			5	8				1
	6		2		1		3	
4				9	6			
9					8	3		
			4	2	9	8		
6	2				3			

Puzzle 16 (Very hard, difficulty rating 0.84)

	3		7		6	9		
1								2
		6	9	2	1			
2			6		3			8
		5		1		7		
3			5		8			9
			2	6	7	4		
6								1
		8	1		4		9	

Puzzle 17 (Hard, difficulty rating 0.60)

3			1		2		7	
			3				6	9
	4	5		6	7		1	
	8	2						
		6				9		
						6	2	
	2		5	8		7	4	
7	5				9			
	1		7		3			6

Puzzle 18 (Medium, difficulty rating 0.48)

3	7	5		6		4		
	6			9			2	
			7					3
4					7		1	
7		6				8		5
	2		3					4
6					1			
	8			3			7	
		4		7		2	5	1

Puzzle 19 (Medium, difficulty rating 0.48)

		6			8	2		
	1			3		9		
				1	4			5
		1		6		7		2
	7		8		2		6	
4		2		7		5		
7			1	8				
		9		4			1	
		4	9			6		

Puzzle 20 (Hard, difficulty rating 0.65)

7					3	1		
		4	5					7
	1	3		4			5	
	5			8	9			
8		9		1		5		3
			3	5			9	
	7			6		4	3	
4					2	7		
		2	4					8

Puzzle 21 (Easy, difficulty rating 0.40)

		2		8				9
9			7			1		6
	8	7	3	6				
		9			6			3
8								2
3			1			6		
			1	8	5	6		
7		1		5				4
2			4		3			

Puzzle 22 (Hard, difficulty rating 0.61)

1		5						
				9		5	3	6
			3					7
	8				4	3	7	
6		3	7		8	4		9
	7	4	5				8	
3				9				
5	2	9		8				
						9		2

Puzzle 23 (Easy, difficulty rating 0.45)

			4	6		8		
						7		3
6		7			8		5	
	5	4	7	2			8	
3								5
	6			4	5	2	9	
	7		5			6		9
2		6						
		3		7	2			

Puzzle 24 (Hard, difficulty rating 0.60)

				3				8
	7	5			4			6
6		1		7			4	
8			7		9		3	
			1		5			
	9		4		3			2
	5			1		2		3
9			3			6	5	
1				5				

Generated by http://www.opensky.ca/~jdhildeb/software/sudokugen/ on Sat Oct 18 07:07:46 2008 GMT. Enjoy!

Puzzle 25 (Medium, difficulty rating 0.56)

			3					7
2	9				6			
1			8		4		9	5
9							1	3
	1	2				7	4	
5	7							8
7	3		9		2			1
			4				3	2
8					5			

Puzzle 26 (Medium, difficulty rating 0.50)

9	1					5		4
	5			1			3	
	6		2		8			9
4		5	3				6	
	7				1	9		3
6			9		4		2	
	4			3			7	
5		7					9	1

Puzzle 27 (Medium, difficulty rating 0.47)

5					8			7
			4			6	8	
						4		9
	8	3		7	4	5		2
	5			8			6	
4		6	3	5		9	7	
1		5						
	6	4			5			
8			2					1

Puzzle 28 (Medium, difficulty rating 0.46)

				2		1		5
	5				7		8	
4		8		6				7
	2		8		6			
	6	9				8	7	
			7		3		1	
5				7		9		1
	7		9				4	
1		4		5				

Puzzle 29 (Easy, difficulty rating 0.42)

1				4				5
			2	5			8	
	3	8	7	6		2		
		7				1	5	
		5				4		
	1	6				3		
		4		3	7	6	2	
	6			2	4			
8				9				7

Puzzle 30 (Easy, difficulty rating 0.28)

9		4		6				7
5					1	4		
		3			5	6	8	
3			9					
8	5						1	6
				2				5
	4	9	5			2		
		7	6					3
2				7		1		4

Puzzle 31 (Easy, difficulty rating 0.40)

8		6		1	9			
	5			6	2			
		9				8	1	
2			9	4		7	8	
	4	8		7	6			1
	7	3				1		
			4	3			7	
			2	8		3		4

Puzzle 32 (Easy, difficulty rating 0.44)

5					1	7		6
		7		8			3	
8			7				5	
6		9	1					
		8	3	7	6	5		
					9	1		8
	9				7			2
	1			9		8		
3		5	2					7

Puzzle 33 (Easy, difficulty rating 0.43)

5							2	
			5	6	9	7	8	3
				8		4	5	
		7	8			9		
			4		5			
		1			7	5		
	9	8		1				
6	3	5	7	4	2			
	4							6

Puzzle 34 (Easy, difficulty rating 0.38)

3		4		1	2		6	
	5	6		8				3
		5				8		1
9			4					7
			9					
7					6			8
6		9		3				
4			9		1	5		
	1		8	6		2		9

Puzzle 35 (Easy, difficulty rating 0.41)

		2			4		3	6
		3			1			7
1			9					
			4	1	9		8	3
		4				2		
6	9		3	2	8			
					7			5
8			5			3		
7	1		6			9		

Puzzle 36 (Very hard, difficulty rating 0.86)

								6
5	8		9					3
		7		4		8	5	
4	2		7	3		9		
		3				7		
		8		1	4		6	2
	7	2		6		5		
8					3		9	7
1								

Puzzle 37 (Easy, difficulty rating 0.44)

	8	2			6			
3		9				7	1	
			4	1				2
6		4		7			8	
	9						4	
	3			9		6		7
5				4	3			
	2	8				9		3
			2			5	7	

Puzzle 38 (Easy, difficulty rating 0.45)

2					7	9	4	
							1	
	6		3	9			2	
3		9	4	8		2		
8				6				7
		4		2	1	8		9
	1			3	8		9	
		3						
	8	6	2					5

Puzzle 39 (Hard, difficulty rating 0.67)

		1	6				9	3
				3		2		
	6	3			4			
6				4	5	8	2	1
8	3	5	1	7				6
			4			5	8	
		2		6				
7	4				3	6		

Puzzle 40 (Hard, difficulty rating 0.61)

		7		5			8	9
4	1			8	7			
8			9				3	
1				4		9		
	7			6			2	
		4		9				5
	9				8			7
		5	1				9	3
6	3			7		2		

Puzzle 41 (Medium, difficulty rating 0.50)

		4		5				
3	7							
5	9		8	1	2			
6		9					1	4
	1		3		9		6	
7	4					9		3
			1	7	6		9	2
							3	8
			3		4			

Puzzle 42 (Medium, difficulty rating 0.48)

6	3				4		1	
4				1			3	
	1					6		
5	2		8				4	
	4		5	2	3		8	
	6				1		2	9
		8					5	
	9			8				7
	5		2				9	1

Puzzle 43 (Easy, difficulty rating 0.35)

					6	9	2	
8							4		3
		6		4	2		7	5
			4	7				
3	7			9			8	6
				5	3			
1	4		9	2		6		
7			3					4
	9	8	1					

Puzzle 44 (Very hard, difficulty rating 0.78)

3		5			8			4
			4	5			1	
1		8			7	6		5
	7	2						3
				2				
5						4	8	
7		6	3			8		1
	9			7	1			
8			5			9		7

Puzzle 45 (Medium, difficulty rating 0.53)

2			8	7		3		
			9			8	6	
	4	8		1		9	7	
	3				9			
9				6				4
			2				1	
	2	3		9		4	5	
	7	9			8			
		5		4	2			3

Puzzle 46 (Easy, difficulty rating 0.35)

8			6	7		3		
					3			
3	6				5	2	8	
1		6				9		
		3	8	2	6	4		
		2				6		5
	1	8	4				9	3
		9						
		5		6	7			1

Puzzle 47 (Medium, difficulty rating 0.48)

8		5				7	6	
								5
				1	2			3
		1	6	9				2
7		2	4		5	9		6
4				7	1	3		
1			7	2				
3								
	9	8				6		7

Puzzle 48 (Hard, difficulty rating 0.60)

						7		
			8			3	5	4
			5			8		1
2		7	4		5	6		
	9	4		1		5	7	
		5	6		8	4		2
9		8			6			
5	1	6			4			
		2						

Generated by http://www.opensky.ca/~jdhildeb/software/sudokugen/ on Sat Oct 18 07:07:47 2008 GMT. Enjoy!

Puzzle 1 (Medium, difficulty rating 0.55)

3	4	1	7	2	5	6	9	8
6	5	9	3	1	8	2	4	7
7	8	2	9	4	6	5	1	3
2	6	5	8	7	4	9	3	1
9	3	8	2	6	1	4	7	5
1	7	4	5	9	3	8	6	2
5	2	7	4	3	9	1	8	6
8	9	6	1	5	7	3	2	4
4	1	3	6	8	2	7	5	9

Puzzle 2 (Easy, difficulty rating 0.39)

5	3	4	9	6	7	2	8	1
6	2	7	8	5	1	9	3	4
1	9	8	2	4	3	7	6	5
4	6	2	7	3	9	1	5	8
9	7	1	4	8	5	3	2	6
8	5	3	1	2	6	4	7	9
7	4	6	3	1	8	5	9	2
2	8	9	5	7	4	6	1	3
3	1	5	6	9	2	8	4	7

Puzzle 3 (Hard, difficulty rating 0.75)

9	5	2	8	7	4	1	3	6
4	6	3	2	5	1	8	9	7
1	7	8	9	3	6	5	4	2
5	2	7	4	1	9	6	8	3
3	8	4	7	6	5	2	1	9
6	9	1	3	2	8	7	5	4
2	3	9	1	8	7	4	6	5
8	4	6	5	9	2	3	7	1
7	1	5	6	4	3	9	2	8

Puzzle 4 (Medium, difficulty rating 0.49)

3	5	8	9	6	1	2	4	7
7	6	9	4	5	2	3	1	8
2	4	1	3	8	7	5	6	9
5	2	3	6	7	4	9	8	1
1	7	4	2	9	8	6	3	5
8	9	6	1	3	5	7	2	4
9	8	2	5	4	6	1	7	3
6	3	7	8	1	9	4	5	2
4	1	5	7	2	3	8	9	6

Puzzle 5 (Hard, difficulty rating 0.61)

6	9	1	2	3	4	7	5	8
8	7	5	1	6	9	4	2	3
2	3	4	8	5	7	9	1	6
9	1	3	5	7	6	8	4	2
4	8	7	9	2	3	5	6	1
5	2	6	4	1	8	3	9	7
1	4	9	3	8	2	6	7	5
7	5	8	6	9	1	2	3	4
3	6	2	7	4	5	1	8	9

Puzzle 6 (Medium, difficulty rating 0.53)

7	9	4	1	3	2	6	5	8
2	5	3	8	7	6	4	1	9
8	6	1	9	5	4	2	7	3
9	4	6	3	8	5	7	2	1
5	1	7	4	2	9	8	3	6
3	2	8	6	1	7	9	4	5
4	8	5	7	6	3	1	9	2
1	3	9	2	4	8	5	6	7
6	7	2	5	9	1	3	8	4

Puzzle 7 (Medium, difficulty rating 0.47)

9	1	8	3	5	4	7	2	6
3	4	5	6	7	2	1	8	9
2	6	7	1	8	9	3	4	5
1	8	4	7	3	5	9	6	2
5	3	2	9	6	8	4	7	1
7	9	6	2	4	1	5	3	8
6	5	3	8	9	7	2	1	4
8	2	9	4	1	3	6	5	7
4	7	1	5	2	6	8	9	3

Puzzle 8 (Hard, difficulty rating 0.71)

4	2	1	8	3	9	6	7	5
6	3	7	1	5	2	9	8	4
5	8	9	6	7	4	3	1	2
7	9	8	3	1	5	2	4	6
3	4	2	9	6	7	8	5	1
1	5	6	2	4	8	7	3	9
8	1	3	4	2	6	5	9	7
2	7	4	5	9	3	1	6	8
9	6	5	7	8	1	4	2	3

Puzzle 9 (Easy, difficulty rating 0.35)

5	9	4	7	2	6	1	3	8
1	8	3	9	5	4	2	7	6
7	6	2	1	3	8	4	9	5
6	7	9	3	1	5	8	2	4
3	1	8	4	9	2	6	5	7
2	4	5	8	6	7	9	1	3
9	5	1	6	4	3	7	8	2
4	2	7	5	8	9	3	6	1
8	3	6	2	7	1	5	4	9

Puzzle 10 (Easy, difficulty rating 0.43)

6	3	4	9	5	7	1	8	2
9	8	1	6	2	4	5	7	3
5	7	2	1	8	3	6	9	4
7	2	5	4	3	9	8	6	1
1	9	6	8	7	2	4	3	5
3	4	8	5	6	1	7	2	9
4	1	7	3	9	8	2	5	6
2	5	9	7	1	6	3	4	8
8	6	3	2	4	5	9	1	7

Puzzle 11 (Medium, difficulty rating 0.56)

6	2	4	3	9	8	5	1	7
5	8	7	1	6	4	9	2	3
9	1	3	7	5	2	6	8	4
1	3	8	2	4	9	7	6	5
7	5	6	8	3	1	4	9	2
4	9	2	5	7	6	1	3	8
2	7	9	4	1	3	8	5	6
3	6	5	9	8	7	2	4	1
8	4	1	6	2	5	3	7	9

Puzzle 12 (Easy, difficulty rating 0.35)

5	4	3	7	6	9	2	8	1
7	1	6	2	5	8	3	9	4
9	8	2	4	3	1	7	6	5
3	5	8	1	4	2	6	7	9
1	9	7	6	8	5	4	2	3
6	2	4	3	9	7	5	1	8
4	3	1	8	7	6	9	5	2
8	6	5	9	2	4	1	3	7
2	7	9	5	1	3	8	4	6

Puzzle 13 (Medium, difficulty rating 0.50)

3	1	2	9	7	5	6	8	4
9	7	4	2	6	8	1	5	3
8	6	5	1	3	4	7	2	9
7	2	6	5	1	3	4	9	8
1	5	9	4	8	7	2	3	6
4	3	8	6	2	9	5	1	7
6	8	7	3	5	2	9	4	1
5	9	1	8	4	6	3	7	2
2	4	3	7	9	1	8	6	5

Puzzle 14 (Medium, difficulty rating 0.53)

5	9	6	1	4	7	3	8	2
3	2	8	5	9	6	4	1	7
4	1	7	8	3	2	5	9	6
2	7	1	3	8	9	6	5	4
6	4	5	2	7	1	9	3	8
8	3	9	6	5	4	2	7	1
9	6	2	7	1	3	8	4	5
1	8	4	9	6	5	7	2	3
7	5	3	4	2	8	1	6	9

Puzzle 15 (Easy, difficulty rating 0.37)

8	7	6	4	3	9	2	1	5
1	9	5	6	2	7	8	4	3
3	4	2	8	1	5	7	6	9
2	3	7	5	8	4	6	9	1
5	6	9	2	7	1	4	3	8
4	8	1	3	9	6	5	2	7
9	1	4	7	6	8	3	5	2
7	5	3	1	4	2	9	8	6
6	2	8	9	5	3	1	7	4

Puzzle 16 (Very hard, difficulty rating 0.84)

5	3	2	7	8	6	9	1	4
1	7	9	3	4	5	8	6	2
4	8	6	9	2	1	3	5	7
2	1	7	6	9	3	5	4	8
8	9	5	4	1	2	7	3	6
3	6	4	5	7	8	1	2	9
9	5	1	2	6	7	4	8	3
6	4	3	8	5	9	2	7	1
7	2	8	1	3	4	6	9	5

Puzzle 17 (Hard, difficulty rating 0.60)

3	6	8	1	9	2	4	7	5
2	7	1	3	5	4	8	6	9
9	4	5	8	6	7	3	1	2
5	8	2	9	7	6	1	3	4
4	3	6	2	1	8	9	5	7
1	9	7	4	3	5	6	2	8
6	2	9	5	8	1	7	4	3
7	5	3	6	4	9	2	8	1
8	1	4	7	2	3	5	9	6

Puzzle 18 (Medium, difficulty rating 0.48)

3	7	5	1	6	2	4	8	9
8	6	1	4	9	3	5	2	7
2	4	9	7	8	5	1	6	3
4	9	3	8	5	7	6	1	2
7	1	6	2	4	9	8	3	5
5	2	8	3	1	6	7	9	4
6	5	7	9	2	1	3	4	8
1	8	2	5	3	4	9	7	6
9	3	4	6	7	8	2	5	1

Puzzle 19 (Medium, difficulty rating 0.48)

3	4	6	5	9	8	2	7	1
5	1	8	2	3	7	9	4	6
2	9	7	6	1	4	3	8	5
8	5	1	4	6	9	7	3	2
9	7	3	8	5	2	1	6	4
4	6	2	3	7	1	5	9	8
7	3	5	1	8	6	4	2	9
6	2	9	7	4	5	8	1	3
1	8	4	9	2	3	6	5	7

Puzzle 20 (Hard, difficulty rating 0.65)

7	8	5	6	9	3	1	2	4
9	6	4	5	2	1	3	8	7
2	1	3	8	4	7	6	5	9
3	5	6	7	8	9	2	4	1
8	4	9	2	1	6	5	7	3
1	2	7	3	5	4	8	9	6
5	7	1	9	6	8	4	3	2
4	9	8	1	3	2	7	6	5
6	3	2	4	7	5	9	1	8

Puzzle 21 (Easy, difficulty rating 0.40)

6	4	2	5	8	1	7	3	9
9	3	5	7	2	4	1	8	6
1	8	7	3	6	9	2	4	5
5	2	9	8	7	6	4	1	3
8	1	6	4	5	3	9	7	2
3	7	4	1	9	2	6	5	8
4	9	3	2	1	8	5	6	7
7	6	1	9	3	5	8	2	4
2	5	8	6	4	7	3	9	1

Puzzle 22 (Hard, difficulty rating 0.61)

1	3	5	6	4	7	2	9	8
7	4	2	8	9	1	5	3	6
8	9	6	3	2	5	1	4	7
2	8	1	9	6	4	3	7	5
6	5	3	7	1	8	4	2	9
9	7	4	5	3	2	6	8	1
3	1	7	2	5	9	8	6	4
5	2	9	4	8	6	7	1	3
4	6	8	1	7	3	9	5	2

Puzzle 23 (Easy, difficulty rating 0.45)

9	3	5	4	6	7	8	1	2
8	4	2	9	5	1	7	6	3
6	1	7	2	3	8	9	5	4
1	5	4	7	2	9	3	8	6
3	2	9	8	1	6	4	7	5
7	6	8	3	4	5	2	9	1
4	7	1	5	8	3	6	2	9
2	8	6	1	9	4	5	3	7
5	9	3	6	7	2	1	4	8

Puzzle 24 (Hard, difficulty rating 0.60)

2	4	9	6	3	1	5	7	8
3	7	5	8	9	4	1	2	6
6	8	1	5	7	2	3	4	9
8	1	6	7	2	9	4	3	5
4	3	2	1	8	5	9	6	7
5	9	7	4	6	3	8	1	2
7	5	4	9	1	6	2	8	3
9	2	8	3	4	7	6	5	1
1	6	3	2	5	8	7	9	4

Puzzle 25 (Medium, difficulty rating 0.56)

4	8	5	3	1	9	2	6	7
2	9	3	7	5	6	1	8	4
1	6	7	8	2	4	3	9	5
9	4	8	2	6	7	5	1	3
3	1	2	5	9	8	7	4	6
5	7	6	1	4	3	9	2	8
7	3	4	9	8	2	6	5	1
6	5	9	4	7	1	8	3	2
8	2	1	6	3	5	4	7	9

Puzzle 26 (Medium, difficulty rating 0.50)

9	1	2	7	6	3	5	8	4
7	5	8	4	1	9	6	3	2
3	6	4	2	5	8	7	1	9
4	2	5	3	9	7	1	6	8
1	9	3	8	4	6	2	5	7
8	7	6	5	2	1	9	4	3
6	8	1	9	7	4	3	2	5
2	4	9	1	3	5	8	7	6
5	3	7	6	8	2	4	9	1

Puzzle 27 (Medium, difficulty rating 0.47)

5	4	2	9	6	8	1	3	7
3	9	1	4	2	7	6	8	5
6	7	8	5	3	1	4	2	9
9	8	3	6	7	4	5	1	2
2	5	7	1	8	9	3	6	4
4	1	6	3	5	2	9	7	8
1	2	5	7	9	3	8	4	6
7	6	4	8	1	5	2	9	3
8	3	9	2	4	6	7	5	1

Puzzle 28 (Medium, difficulty rating 0.46)

6	3	7	4	2	8	1	9	5
9	5	2	1	3	7	4	8	6
4	1	8	5	6	9	2	3	7
7	2	1	8	4	6	3	5	9
3	6	9	2	1	5	8	7	4
8	4	5	7	9	3	6	1	2
5	8	3	6	7	4	9	2	1
2	7	6	9	8	1	5	4	3
1	9	4	3	5	2	7	6	8

Puzzle 29 (Easy, difficulty rating 0.42)

1	7	2	3	4	8	9	6	5
6	4	9	2	5	1	7	8	3
5	3	8	7	6	9	2	1	4
4	9	7	6	8	3	1	5	2
3	8	5	9	1	2	4	7	6
2	1	6	4	7	5	3	9	8
9	5	4	8	3	7	6	2	1
7	6	1	5	2	4	8	3	9
8	2	3	1	9	6	5	4	7

Puzzle 30 (Easy, difficulty rating 0.28)

9	1	4	2	6	8	3	5	7
5	6	8	7	3	1	4	2	9
7	2	3	4	9	5	6	8	1
3	7	1	9	5	6	8	4	2
8	5	2	3	4	7	9	1	6
4	9	6	1	8	2	7	3	5
6	4	9	5	1	3	2	7	8
1	8	7	6	2	4	5	9	3
2	3	5	8	7	9	1	6	4

Puzzle 31 (Easy, difficulty rating 0.40)

8	3	6	7	1	9	4	5	2
1	5	4	8	6	2	9	3	7
7	2	9	3	5	4	8	1	6
2	6	1	9	4	3	7	8	5
5	9	7	1	2	8	6	4	3
3	4	8	5	7	6	2	9	1
4	7	3	6	9	5	1	2	8
6	8	2	4	3	1	5	7	9
9	1	5	2	8	7	3	6	4

Puzzle 32 (Easy, difficulty rating 0.44)

5	2	4	9	3	1	7	8	6
9	6	7	4	8	5	2	3	1
8	3	1	7	6	2	9	5	4
6	5	9	1	2	8	4	7	3
1	4	8	3	7	6	5	2	9
2	7	3	5	4	9	1	6	8
4	9	6	8	5	7	3	1	2
7	1	2	6	9	3	8	4	5
3	8	5	2	1	4	6	9	7

Puzzle 33 (Easy, difficulty rating 0.43)

5	8	3	1	7	4	6	2	9
2	1	4	5	6	9	7	8	3
9	7	6	2	8	3	4	5	1
4	5	7	8	3	1	9	6	2
3	6	9	4	2	5	8	1	7
8	2	1	6	9	7	5	3	4
7	9	8	3	1	6	2	4	5
6	3	5	7	4	2	1	9	8
1	4	2	9	5	8	3	7	6

Puzzle 34 (Easy, difficulty rating 0.38)

3	8	4	7	1	2	9	6	5
1	9	5	6	4	8	7	2	3
2	6	7	5	3	9	8	4	1
9	3	2	4	8	5	6	1	7
8	4	6	1	9	7	3	5	2
7	5	1	3	2	6	4	9	8
6	7	9	2	5	3	1	8	4
4	2	8	9	7	1	5	3	6
5	1	3	8	6	4	2	7	9

Puzzle 35 (Easy, difficulty rating 0.41)

9	5	2	8	7	4	1	3	6
4	6	3	2	5	1	8	9	7
1	7	8	9	3	6	5	4	2
5	2	7	4	1	9	6	8	3
3	8	4	7	6	5	2	1	9
6	9	1	3	2	8	7	5	4
2	3	9	1	8	7	4	6	5
8	4	6	5	9	2	3	7	1
7	1	5	6	4	3	9	2	8

Puzzle 36 (Very hard, difficulty rating 0.86)

2	4	9	3	8	5	1	7	6
5	8	6	9	7	1	4	2	3
3	1	7	6	4	2	8	5	9
4	2	1	7	3	6	9	8	5
6	5	3	8	2	9	7	1	4
7	9	8	5	1	4	3	6	2
9	7	2	4	6	8	5	3	1
8	6	4	1	5	3	2	9	7
1	3	5	2	9	7	6	4	8

Puzzle 37 (Easy, difficulty rating 0.44)

1	8	2	7	3	6	4	9	5
3	4	9	5	8	2	7	1	6
7	6	5	4	1	9	8	3	2
6	5	4	3	7	1	2	8	9
8	9	7	6	2	5	3	4	1
2	3	1	8	9	4	6	5	7
5	7	6	9	4	3	1	2	8
4	2	8	1	5	7	9	6	3
9	1	3	2	6	8	5	7	4

Puzzle 38 (Easy, difficulty rating 0.45)

2	3	5	8	1	7	9	4	6
9	4	8	6	5	2	1	7	3
1	6	7	3	9	4	5	2	8
3	7	9	4	8	5	2	6	1
8	2	1	9	6	3	4	5	7
6	5	4	7	2	1	8	3	9
7	1	2	5	3	8	6	9	4
5	9	3	1	4	6	7	8	2
4	8	6	2	7	9	3	1	5

Puzzle 39 (Hard, difficulty rating 0.67)

4	5	1	6	2	8	7	9	3
9	8	7	5	3	1	2	6	4
2	6	3	7	9	4	1	5	8
6	7	9	3	4	5	8	2	1
1	2	4	9	8	6	3	7	5
8	3	5	1	7	2	9	4	6
3	9	6	4	1	7	5	8	2
5	1	2	8	6	9	4	3	7
7	4	8	2	5	3	6	1	9

Puzzle 40 (Hard, difficulty rating 0.61)

3	2	7	1	5	6	4	8	9
4	1	9	3	8	7	5	6	2
8	5	6	9	2	4	7	3	1
1	8	3	2	4	5	9	7	6
9	7	5	8	6	1	3	2	4
2	6	4	7	9	3	8	1	5
5	9	2	6	3	8	1	4	7
7	4	8	5	1	2	6	9	3
6	3	1	4	7	9	2	5	8

Puzzle 41 (Medium, difficulty rating 0.50)

1	2	4	7	5	3	6	8	9
3	7	8	9	6	4	2	5	1
5	9	6	8	1	2	3	4	7
6	3	9	5	2	7	8	1	4
8	1	2	3	4	9	7	6	5
7	4	5	6	8	1	9	2	3
4	8	3	1	7	6	5	9	2
2	6	7	4	9	5	1	3	8
9	5	1	2	3	8	4	7	6

Puzzle 42 (Medium, difficulty rating 0.48)

6	3	5	7	9	4	2	1	8
4	8	7	6	1	2	9	3	5
9	1	2	3	5	8	6	7	4
5	2	1	8	6	9	7	4	3
7	4	9	5	2	3	1	8	6
8	6	3	4	7	1	5	2	9
1	7	8	9	3	6	4	5	2
2	9	4	1	8	5	3	6	7
3	5	6	2	4	7	8	9	1

Puzzle 43 (Easy, difficulty rating 0.35)

4	3	7	5	8	6	9	2	1
8	5	2	7	1	9	4	6	3
9	1	6	3	4	2	8	7	5
5	6	9	4	7	8	3	1	2
3	7	4	2	9	1	5	8	6
2	8	1	6	5	3	7	4	9
1	4	5	9	2	7	6	3	8
7	2	3	8	6	5	1	9	4
6	9	8	1	3	4	2	5	7

Puzzle 44 (Very hard, difficulty rating 0.78)

3	2	5	6	1	8	7	9	4
9	6	7	4	5	3	2	1	8
1	4	8	2	9	7	6	3	5
4	7	2	9	8	5	1	6	3
6	8	3	1	2	4	5	7	9
5	1	9	7	3	6	4	8	2
7	5	6	3	4	9	8	2	1
2	9	4	8	7	1	3	5	6
8	3	1	5	6	2	9	4	7

Puzzle 45 (Medium, difficulty rating 0.53)

2	9	6	8	7	5	3	4	1
3	1	7	9	2	4	8	6	5
5	4	8	3	1	6	9	7	2
6	3	1	4	5	9	2	8	7
9	8	2	7	6	1	5	3	4
7	5	4	2	8	3	6	1	9
1	2	3	6	9	7	4	5	8
4	7	9	5	3	8	1	2	6
8	6	5	1	4	2	7	9	3

Puzzle 46 (Easy, difficulty rating 0.35)

8	2	1	6	7	9	3	5	4
5	4	9	2	8	3	1	7	6
3	6	7	1	4	5	2	8	9
1	7	6	5	3	4	9	2	8
9	5	3	8	2	6	4	1	7
4	8	2	7	9	1	6	3	5
6	1	8	4	5	2	7	9	3
7	3	4	9	1	8	5	6	2
2	9	5	3	6	7	8	4	1

Puzzle 47 (Medium, difficulty rating 0.48)

8	2	5	3	4	9	7	6	1
9	1	3	8	6	7	2	4	5
6	7	4	5	1	2	8	9	3
5	3	1	6	9	8	4	7	2
7	8	2	4	3	5	9	1	6
4	6	9	2	7	1	3	5	8
1	4	6	7	2	3	5	8	9
3	5	7	9	8	6	1	2	4
2	9	8	1	5	4	6	3	7

Puzzle 48 (Hard, difficulty rating 0.60)

8	5	3	1	4	2	7	6	9
7	2	1	8	6	9	3	5	4
4	6	9	5	3	7	8	2	1
2	8	7	4	9	5	6	1	3
6	9	4	2	1	3	5	7	8
1	3	5	6	7	8	4	9	2
9	7	8	3	2	6	1	4	5
5	1	6	9	8	4	2	3	7
3	4	2	7	5	1	9	8	6