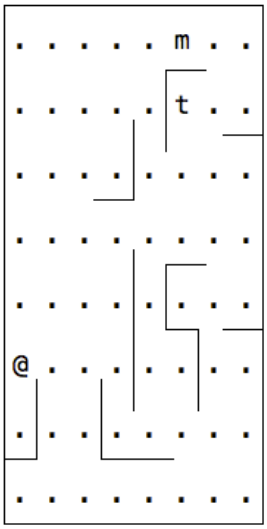
Computer Science 237
Assignment 1
Due next week, before lab.

The purpose of this week's lab is to finish off a program that implements a wonderful *logic maze* by Robert Abbott: Theseus and Minotaur. I have written most of the code to play this solitare game; what remains is the automatic movement of the evil Minotaur. Once this method is written correctly, you should be able to solve each of the 87 levels in turn. This handout describes the task.

Theseus and Minotaur is a maze that reconfigures using a mechanism or a *logic*. You control one player, Theseus, who seeks to make it to the *goal*. The computer controls the Minotaur; he chases Theseus at an inhuman speed. Facing the Minotaur, I needn't mention, is certain death for our hero.

Here, the puzzle shows an overhead view of level 10. Theseus appears as `t`, the Minotaur as `m`, and the goal as `@`. Dots (`.`) represent open space.



Theseus and Minotaur, level 10, at the start.

Using familiar control keys, you can move Theseus in any of the four compass directions, provided a wall is not in the way. If you choose, Theseus can also mark a good spot, sit down, and eat lunch. In response to any move, the Minotaur makes *two* moves using this aggressive strategy:

1. He always moves closer to Theseus.
2. If possible, he moves horizontally (left or right), otherwise
3. He moves vertically (up or down), if possible.

It hardly seems fair.

The puzzle has been written to be played in the traditional way: on a terminal using the symbols shown above. A win occurs when Theseus appears at the goal (`T`) without the Minotaur. If the Minotaur is alone at the goal he appears as `M`. Our hero loses whenever he's overtaken by the Minotaur; this is displayed as `*`, indicative of a *dusty kerfuffle*.

To reenforce your emacs skills, you'll find Theseus can be moved about using the traditional emacs cursor movement functions for line-up (`C-p`), line-down (`C-n`), left (`C-b`) and right (`C-f`). Theseus can pass the time by marking a good spot for lunch (`C-space`). Undo is, as it is in emacs, `C-_`. To give up, type `C-g`.

You can pick up the distribution of this lab from the course directory as `/usr/cs-local/share/cs237/kits/theseus.tar.gz`. This file is 'tarred' and 'zipped'. To unzip and untar this package with

```
tar xvfz theseus.tar.gz
```

The result is a directory, `theseus`, that contains the source, this lab handout, and a number of data files.

To build the program, you simply type:

```
make theseus
```

Whenever you make changes to any of the source files, you simply type the make command again. When you are finished, you may want to

```
make clean
```

which gets rid of all the cruft that is left behind during the development and debugging processes. It leaves the executable alone.

If you want to get rid of the executable as well, you can type:

```
make realclean
```

and the directory is left in a pristine condition.

All I require is that you write one procedure (see `chase.c`),

```
extern int moveM(level *l);
```

This routine is responsible for moving the Minotaur *one step*. This might not be possible (because of walls, for example), so we indicate success by returning a 1, otherwise a 0. At the moment, `moveM` always returns zero, so the Minotaur stands motionless. Still you can lose and win, but life is most interesting when you risk it.

Ideally this procedure gets the positions of Theseus and the Minotaur and makes a decision how to move. Once the decision is made, the screen is updated with the Minotaur in its new location.

Internally, the level is a rectangular array of characters, each of which potentially has one or more of the following bits set:

`LEFT, RIGHT, UP, DOWN`. This location has walls to the left, right, up, or down. They are individually set, in all combinations.

For internal record keeping, other bits may be set as well. They should not have any impact on the interpretation of the contents of the location.

You will find the following methods of help:

- `int get(level *t, int row, int col)`. Returns the character that describes the contents of the cell at that location. We're primarily interested in the four bits masked by `LEFT`, `RIGHT`, `UP`, and `DOWN`.
- `int getT(level *l)`. Gets the location of Theseus. The result is an integer, but only the low two nibbles are set: bits 4-7 contain the row, and bits 0-3 contain the column.
- `int getM(level *l)`. Gets the location of Minotaur. Again, the result is an integer, but only the low two nibbles are set.
- `void movePiece(level *l, int player, int r0, int c0, int r1, int c1)`. Moves a player from one location to another. You should move the `MINOTAUR`. This includes code that manages the undo stack.

There are many other methods. I would suggest you not make use of them in this assignment. The reasoning is simple: next week we will be converting your `chase.c` methods to assembly code and you will want to limit the amount of code you convert.

Otherwise, you are free to edit the `theseus` code or add new levels.

Please: place your code in a file called `chase.c` and turn it in using `turnin -c 237 chase.c` before next week's lab.