Hitori

Assignment 4: Heuristics and basic grid generation

The objective of this assignment is to develop heuristics that will help solving the grid without stupidly testing all black/white possibilities. The other objective is to generate grids to be able to test your code.

A non-exhaustive list of heuristics is given below:

- When one number is sandwiched by two different numbers, but identical to each other, the middle number must be marked as `white`. Since one of the outside two numbers must be `black`, the inside number must be `white` or there would be an adjacent `black` cell.
- If there are two consecutive cells of same number in a row/column and the row/column contains another identical number, that number must be `black`. If not, then both the adjacent cells would have to be `black`.

Furthermore:

- When a cell is marked as `black`, all adjacent cells can be immediately marked as `white`, since no adjacent cells may be `black`.
- If a cell is determined to be `white`, any identical digits in the row/column must be `black`.

2. Development and application of heuristics

1. Create new files `include/heuristics.h` and `src/heuristics.c`
2. Write a function for each heuristic (you may invent new ones!). The function for a given heuristic must:
    a. Take as parameter a pointer to a grid
    b. Return a Boolean value: `true` if applying the heuristic changed the grid, `false` otherwise
3. Write a function that repetitively applies the heuristics until stability (meaning that no heuristic changes the grid anymore).

3. Basic grid generation and printing

1. Create new files `include/generate.h` and `src/generate.c` that contain grid generation code
2. Write a function that generates a new grid with random values (implementation of command line argument `-g`). The grid to be generated will be allocated by the caller of the function.

   For now, the grid generation function will be basic (generation of cells with randomly generated values that are significant depending of grid size). Therefore, at this point automatically generated grids may not have any solution. The interface of the grid allocation function will be the following:

   ```
   void grid_generate(grid_t *g);
   ```

4. Testing

1. Automate testing by preparing a set of grids that will be placed in the `tests/` directory
2. Heavily test your code on your own grids and the grids given to you by your professors