

Usage

1. Compilation

```
cc -Wall -Wextra -Werror *.c -o skyscraper
```

2. Running

The program expects **one string argument** containing 16 numbers (1–4) separated by spaces. These numbers represent the border clues in this order:

TOP(4 →) BOTTOM(4 →) LEFT(4 ↓) RIGHT(4 ↓)

"col1top col2top col3top col4top col1bottom col2bottom col3bottom col4bottom row1left row2left row3left row4left row1right row2right row3right row4right"



Example:

```
./skyscraper "2 1 4 2 2 3 1 3 2 3 1 2 2 1 3 2"
```



Example Output

```
2 4 1 3
1 3 2 4
4 1 3 2
3 2 4 1
```

If no solution exists or the input is not valid, the program prints:

Error



Project Structure

- **skyscraper.c** → entry point, solver execution, backtracking algorithm (`main`, `solve`, `field_is_a_solution`)
- **validations.c** → input and rule validation functions
- **get_arrays.c** → row/column generation (`permute`, `get_arrays`)
- **helpers.c**, **field_helpers.c**, **get_arrays_helpers.c** → utility functions and argument parsing (`swap`, `reverse`, `display_field`, `grab_ints_from_str` etc.)



Features

- Clear separation of concerns (input validation, row/column generation, solver).
- Uses **backtracking with pruning** → efficient search.
- Prints either the **solution grid** or **Error**.
- Full compliance with **Norminette** (no long functions, 4 arguments maximum each, etc.).
- Only `write()`, `malloc()`, and `free()` were allowed to be used