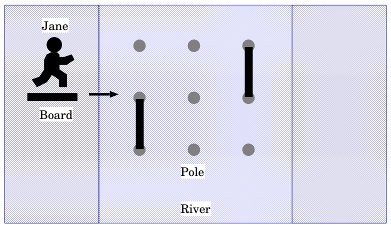
ECE368 Programming Assignment #6

Due Friday, April 28, 2017, 11:59pm

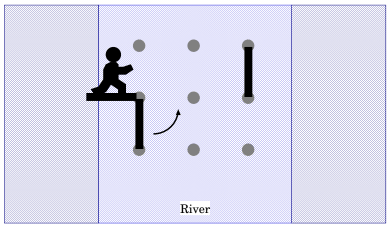
Description

This project is to be completed on your own. This problem is made available by Prof. Martin Henz from the Computer Science Department of the National University of Singapore.

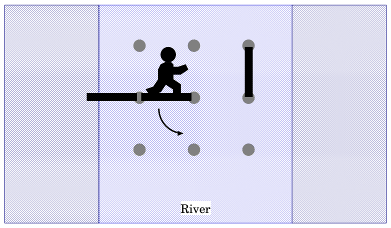
Jane wants to cross a river. The river is dotted with $N \times M$ poles to which boards can latch on. Some of the poles already have boards latched on such that the boards point in the direction of the river flow (from top to bottom). For $N = 3$ and $M = 3$, the situation may look like this.



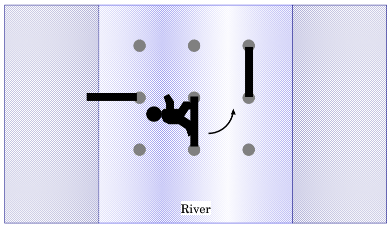
On her side of the river, Jane has a board that points to the other side. She can latch the board on any of the first column of poles, as shown in (1). She can then use the board as a bridge to reach the pole the board is latched on. When Jane is on a board, she can unlatch one of the two ends of the board and turn the board by 90 degrees such that it latches on another pole, as shown in (1)→(2). Jane can also turn a board by 180 degrees by turning it by 90 degrees twice, as shown in (2)→(3) and (3)→(4). Since turning a board is quite tiresome, she wants to cross the river in such a way that she needs to turn boards by 90 degrees as few times as possible. She does not require any turning to reach any of the first column of poles as she can simply push the board to latch on any of the first column of poles. (However, she may have to turn that board in order to reach a pole in the second column.) In this particular configuration of poles and boards, Jane can cross the river with 4 board turns (5).



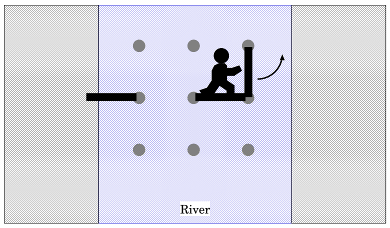
(1)



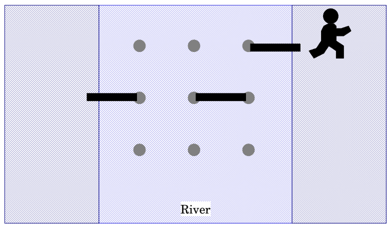
(2)



(3)



(4)



(5)

Deliverables:

In this project, you are to develop your own include file `river.h` and source files `river.c` and `river_main.c`, which can be compiled with the following command:

```
gcc -Werror -Wall -Wshadow -O3 river.c river_main.c -o proj6
```

All declarations and definition of data structures and functions must reside in the include file or source file. The executable `proj6` would be invoked as follows:

```
proj6 input_file
```

The executable loads the configuration of the poles and boards from `input_file`, calculates the number of turns required for Jane to cross the river, and output the number of turns to the `stdout`.

The input file, in text form, has the following format. The first line contains two numbers (separated by a space and the line is terminated with a newline character). The first number is the number of rows of poles, and the second number is the number of columns of poles. Let N be the number of rows and M be the number poles. Subsequently, the input file has $N-1$ lines. Each of the $N-1$ lines contains M characters (not counting the newline character at the end of the line), which are 0 or 1, and these M characters describe the board configuration between two adjacent rows of poles. The character 1 means there is a board and the character 0 means there is no board. Thus the second line of the input file describes the board configuration between the top two rows of poles, and so on, and the last input line describes the board configuration between the bottom two rows of poles. The first of the N characters describes the leftmost board configuration between the leftmost two poles of two adjacent rows of poles, and so on, and the rightmost character describes the rightmost board configuration between the rightmost two poles of the same two rows of poles.

The input file for the starting configuration of the given example contains:

```
3 3
001
100
```

The output to the stdout should be simply the number 4 followed by a newline (use `"%d\n"` as the formatting string in the `printf` function).

Submission:

The project requires the submission (through Blackboard) of a zip file containing the source and include files and a report detailing the results. The report should describe the data structure and the approach you have used to solve the problem. You should also comment on the run-time complexity of your algorithm (for the construction of the chosen data structure to represent the input and also for the computation of the number of turns required). Your report should not be longer than 1 page and should be in plain text format or pdf.

You should create a zip file called proj6.zip that contains all of the above and submit the zip file. If the zip file contains a makefile, we will use that file to make your executable.

Grading:

The grade depends on the correctness of your program, the efficiency of your program, the clarity of your program documentation and report.

The report will account for 10% of the entire grade. The program will account for 90% of the entire grade.

It is important all the files that have been opened are closed and all the memory that have been allocated are freed before the program exits. Memory errors will result in at least 50% penalty.

Getting started:

We provide a sample input files (in `samples.zip`) for you to evaluate your program. The input files are `river.in[0-3]`, and files storing the corresponding screen output are `river.log[0-3]`.

The example provided in this description can be found in `river.in0`.

Copy over the files from the Blackboard website. Check out the Blackboard website for any updates to these instructions.