The University of Queensland
School of Information Technology and Electrical Engineering
Semester Two, 2016
CSSE2310 / CSSE7231 - Assignment 1

Introduction

Your task is to write an C99 program (called `nogo`) which allows the user to play a game of atari-go(described later). This will require I/O from both the user and from files. Your assignment submission must comply with the C style guide (v2.0.2) available on the course website.

The Game

`nogo` will display a grid of cells like this:

```
/----\
|....|
|....|
|....|
|....|
\----/
```

The dots indicate empty cells (the other marks are just borders). The two s in the game will take turns putting "stones" into empty cells. The first will indicate their stones with `O` and the second will use `X`. The cells are numbered from the top left corner as 0,0 (row, then column).

Connected sets of stones of the same type form "strings". To be connected, stones need to be horizontally or vertically adjacent. In this example:

```
/----\
|OOO.|
|...O|
|.X..|
|X.X.|
\----/
```

The first player has 2 strings while the second player has 3.

Strings may have "liberties" (spaces which they could grow into). A liberty is an empty cell horizontally or vertically adjacent to a string. In this example:

```
/----\
|X.OX|
|..OO|
|..X.|
|....|
\----/
```

The `X` at 0,3 has no liberties; the `X` at 2,2 has 3 liberties and the string of Os has 3.

A string is considered "captured" when it has no liberties. So in the example above, the string at 0,3 would be captured. The first player to have one of their strings captured loses. In cases where both s would have a string captured for example:

```
/-----\
|OX OX|
|OOXXX|
|O....|
|.....|
\-----/
```

the player placing the stone checks their opponent's strings first. So, if O placed at 0,2 then X would lose.

Interaction

At the beginning of the game and after each player's turn, the grid will be displayed. There are two types of in the game, `h` (where moves will be read from `stdin`) and `c` (where moves will be generated by your program). For `c`-s, the move the program generates will be printed like this (substitute X for O as needed):
`Player O: 1 0`
then go to the next player's turn. For `h`-s, the user will be prompted:
`Player X>`
and wait for the user to enter *row column* (single space separated with no leading nor trailing spaces). If the user's input is not valid (eg not two numbers, not within the bounds of the grid, not empty), then the prompt is reprinted for them to try again.

When a winner is determined, the following is displayed:

`Player ? wins`

(substitute O/X for ? as appropriate).

Invocation

When run with an incorrect number of arguments, `nogo` should print usage instructions to stderr:

```
Usage: nogo p1type p2type [height width | filename]
```

and exit (see the error table).

`p1type` and `p2type` must be either `c` or `h`. `height` and `width` (measured in cells) must be integers between 4 and $1,000$ inclusive. If height and width are omitted, then a filename can be given instead, in that case, the program should try to load a saved game from the named file.

Saved game file format

Saved game files will consist of a line of 9 numbers (separated by single spaces) followed by the contents of the grid (without border markings). The last line of the file must be terminated with newline. For example:

```
5 5 1 4 1 4 2 0 0
XX...
....O
.....
O....
O....
```

The numbers are (in order):

- height of the grid
- width of the grid
- 0 or 1 to indicate which player is next to play. 1 indicates X is to play next.
- the next two numbers indicate the row and column a computer for `O` will try.
- the number of the next move for the computer for O.
- the next two numbers are the next row and column for a computer for `X` will try.
- the number of the next move for the computer for X.

Note that the last number is zero, but X has made 2 moves already. This is because in this example, Player X was a human and so did not use the computer . Things to watch out for:

- Files with fewer lines than the grid size requires.
- Files with short lines (less content than expected).
- Files with lines which are longer than expected.
- Files with characters in the wrong places.
- Next player being something other than 0 or 1.

Computer moves

The computer s will generate their moves using the following algorithm (note that the should continue generating moves until an empty space is found):

- I_r is the initial row (1 for O and 2 for X).
- I_c is the initial column (4 for O and 10 for X).
- F is a multiplication factor (29 for O and 17 for X).
- G_w is the width of the grid.
- G_h is the height of the grid.
- M is a counter of moves generated (starting at zero).

Initial setup:
$r = I_r, c = I_c$
$B = I_r * G_w + I_c$
To get a move, return (r mod G_h) and (c mod G_w).
To generate the next move.

- if M is a multiple of 5:
  - $N = (B + M/5 * F)$ mod 1000003
  - $r = N/G_w$ (integer division)

- $c = N \bmod G_w$

- if $M \bmod 5$ is:

  1. $r+ = 1$, $c+ = 1$
  2. $r+ = 2$, $c+ = 1$
  3. $r+ = 1$, $c+ = 0$
  4. $r+ = 0$, $c+ = 1$

So for a 7×7 grid the move sequence generated for Player O would start with:
$(1,4), (2,5), (4,6), (5,6), (5,0)$
$(5,5), (6,6), (1,0), (2,0), (2,1)$
$(2,6), (3,0), (5,1), \ldots$

Saving games

To save a game, instead of entering a row and column, enter `w` followed immediately by the path of the file to save to. That is, no space between `w` and the start of the path.

Errors and messages

When one of the conditions in the following table happens, the program should print the error message and exit with the specified status. All error messages in this program should be sent to standard error and followed by a newline. Error conditions should be tested in the order given in the table.

Condition	**Exit Status**	**Message**
Program started with incorrect number of arguments	1	Usage: nogo p1type p2type [height width | filename]
Incorrect types	2	Invalid type
Board dimensions invalid	3	Invalid board dimension
Can't open save file for reading	4	Unable to open file
Error reading game. Eg: bad chars in input, not enough lines, short lines	5	Incorrect file contents
End of file while waiting for user input	6	End of input from user

For a normal exit, the status will be 0 and no special message is printed.

There are a number of conditions which should cause messages to be displayed but which should not immediately terminate the program. These messages should also go to standard error.

Condition	**Action**	**Message**
Error opening file for saving grid	Prompt again	Unable to save game

Compilation

Your code must compile with command:
`make`
When you compile, you must use at least the following flags: `-Wall -pedantic -std=gnu99`.

You must not use flags or pragmas to try to disable or hide warnings.

Submission

Submission must be made electronically by committing using subversion.

Example game — two computer s

```
/----\
|....|
|....|
|....|
|....|
\----/
Player O: 1 0
/----\
|....|
|O...|
|....|
|....|
\----/
Player X: 2 2
/----\
|....|
|O...|
|..X.|
|....|
\----/
Player O: 2 1
/----\
|....|
|O...|
|.OX.|
|....|
\----/
Player X: 3 3
/----\
|....|
|O...|
|.OX.|
|...X|
\----/
Player O: 0 2
/----\
|..O.|
|O...|
|.OX.|
|...X|
\----/
Player X: 2 0
/----\
|..O.|
|O...|
|XOX.|
|...X|
\----/
Player O: 1 2
/----\
|..O.|
|O.O.|
|XOX.|
|...X|
\----/
Player X: 0 3
/----\
|..OX|
|O.O.|
|XOX.|
|...X|
\----/
Player O: 1 3
/----\
|..OX|
|O.OO|
|XOX.|
|...X|
\----/
Player O wins
```