CS 214: Introduction to computer systems
Winter Term, 2026

Lab 3

In this lab, you'll begin working with C, using Connect Four as a motivating application.

Connect Four

For this lab, you'll be using `euclid`. We'll be connecting to this machine using the ssh protocol, which establishes an encrypted connection.

To connect, you need to open an ssh connection. On a Mac, start `Terminal` (it's in Applications → Utilities) and type the command `ssh` *username*`@euclid`. You'll be prompted for your password (it's your Knox password and there isn't any display that you're typing in the password (i.e. no asterisks or whatever to show the number of characters)). On Windows, you now seem to be able to type the ssh command into a Terminal; if not, you can download the program putty (first hit if you Google `putty`). Run this and connect to the host `euclid`. You'll be prompted for your (Knox) username and password.

Then open your (initially-blank) file `connect4.c` (with `emacs connect4.c`). Once you have opened the file, you can create a `board` variable that is a 2D array (8×8) of characters by putting the following in the file:

```
char board[8][8];
```

Next create a `main` function for the program to run. Recall that the simplest `main` is the following:

```
int main() {
}
```

Once you've typed this in and saved the file (Control-x Control-s), you can suspend emacs with Control-z to get back to a prompt. Then type the following to compile and run this program:

```
gcc -Wall -std=gnu99 -o connect4 connect4.c
./connect4
```

Once you run that (and see it does nothing...), reopen the suspended version of emacs using the command `fg`. Inside the `main` function, add a nested loop to set every cell of the 2D array to contain a period:

```
for(int i=0; i < 8; i++)
  for(int j=0; j < 8; j++)
    board[i][j] = '.';
```

(Unlike in Java, C does not automatically initialize your variables for you.) Then write a second nested loop to print the contents of this variable row by row. Make `board[0][0]` the bottom left coordinate, `board[7][0]` the bottom right coordinate, etc; the first array index is the x coordinate and both coordinates count from the bottom left. You can check your code by temporarily setting the corners to distinctive values before printing the board:

```
board[0][0] = 'A';
board[0][7] = 'B';
board[7][0] = 'C';
board[7][7] = 'D';
```

I suggest that you put the printing code into a function.

Once you have the board printing, write code so that the red player can make a move. This means they enter an integer in the range 0–7 and the lowest period in that column is replaced with an 'R' character. To read a value into the integer variable `x`, use the following:

```
scanf("%d", &x);
```

(Remember that you'll need to add `#include <stdio.h>` to the head of your file to use `scanf`.)

Next, modify your program so that after the red player moves, the blue player gets a turn (putting a 'B' into the lowest period in some column), followed by the red player again, with the players alternating until the entire board is full. Again, it is helpful if the code for making a move is a function that takes the player (either 'R' or 'B') as an argument.

Once you can make moves, it is time to start detecting winning moves. A player wins if they ever get 4 of their pieces into consecutive positions, i.e. 4 pieces next to each other in a row, column, or diagonal line. If this happens, print a message about that player winning and exit the program. (You can use the function `exit` to exit the program. You should pass it the argument 0 and will need to add `#include <stdlib.h>` to the top of the file.)

If you get all of that working, spend the rest of the period refining the program. (At the end of the period, be sure to save your program again and exit emacs with Control-x Control-c.) Can you detect illegal moves (out of range values or trying to put another piece into a column that is already full)? If a player tries one of these things, refuse the move and ask for another. What about allowing rematches and keeping a running score of games won? Suggesting winning moves or moves that block the other player from winning moves?