CS106B
Autumn 2012

Handout 17Y
October 17, 2012

YEAH Outline: Assignment 3

Will Monroe

Logistix

Recursion warmup problems: due Monday, 10/22, 3pm
Boggle: due Friday, 10/26, 3pm
Late days are counted separately for each
Honestly, though, using a late day for the warmups is a Very Silly Idea.
    recommendation: pretend they're due Fri., so you'll have time for Boggle.

Motivation: Recursion

Fact: any problem that can be solved with recursion can be solved with iteration.
    [Alonzo Church and Alan Turing—2 names every programmer should know!]
So why recursion? The answer, in truth, is *style*.
The iterative solution might work, but for some problems it's long, messy, hard to understand, and even harder to debug.
Meanwhile, recursive solutions for these problems can be works of art: short, simple, and obvious in retrospect (although still not easy to come up with!)

The general structure of a recursive solution

1. If the problem is simple enough (a “base case”), just return the answer.
2. Otherwise, split it into one *or more* smaller but similar subproblems.
3. Solve each subproblem, assuming your solution works (the “leap of faith”).
4. *Combine* the answers for each subproblem to arrive at a final answer.

Mental block number 1: forgetting you might need more than one subproblem.
    You could have several recursive calls, maybe (in fact, usually) in a `for` loop.
Mental block number 2: not considering how to combine the sub-answers.
    This step is sometimes so easy it's dumb, but don't forget it's there.
    e.g. recursive backtracking: your sub-answers are `bool`s (can/can't solve), and you combine them by returning `true` if any are `true`, `false` if none are

Warmup: Trees

[The bane of a video game developer's life. It's hard to make a convincing tree!]
Graphics:
```
(GWindow &window)
GPoint end = window.drawPolarLine(start, distance, angle);
```

Warmup: Dominosa

Valid solutions are always verticals plus parallel *pairs* of horizontals. (Play with the demo to convince yourself you needn't worry about catty-corner horizontals.)
Graphics:
```
DominosaDisplay display;
display.drawBoard(board);
display.provisonallyPair(firstCoord, secondCoord); [sic]
display.certifyPairing(firstCoord, secondCoord);
display.vetoProvisionalPairing(firstCoord, secondCoord);
display.eraseProvisionalPairing(firstCoord, secondCoord);
```

Boggle!
A classic example of a problem that is doable with recursion, but awful without it.

The handout warns you not to try to combine the human and computer turns.
Another thing you should be wary of combining: finding the cube to *start* at,
versus finding the *next* cube from your current position (in both turns)
Yes, this means a lot of double `for` loops. Decompose from the start! Otherwise,
indentation, loop variable names, *etc*., will drive you nuts.

If you modify objects passed by reference in your recursive backtracking,
remember to clean up after yourself before returning false!

Graphics:

```
initGBoggle(window);
drawBoard(numRows, numCols);
labelCube(row, col, letter);
highlightCube(row, col, isHighlighted);
recordWordForPlayer(word, player);
```

And another very useful function: `if(grid.inBounds(row, col)) ...`
(hopefully you found this for Life. If not, you get another chance!)

Questions? Answers.

—in case it was unclear, Big Boggle (5x5) is optional and will be worth a small
amount of extra credit. Think about other cool things you could do!