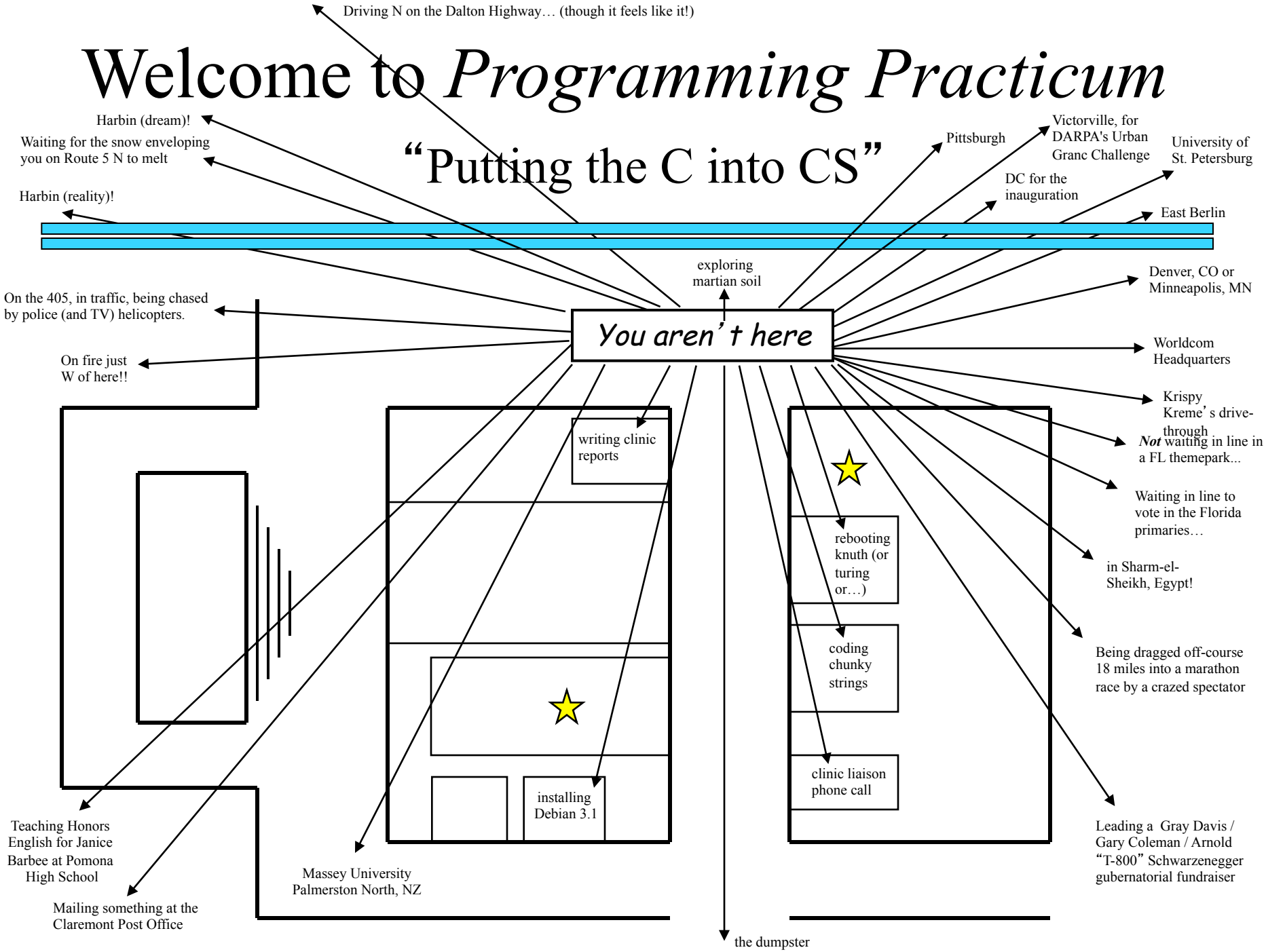
Welcome to *Programming Practicum*

"Putting the C into CS"



Introductions!

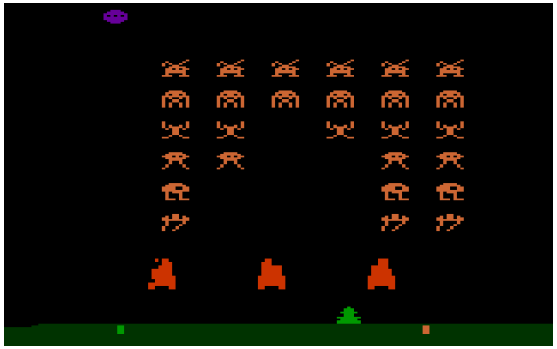
Zach Dodds

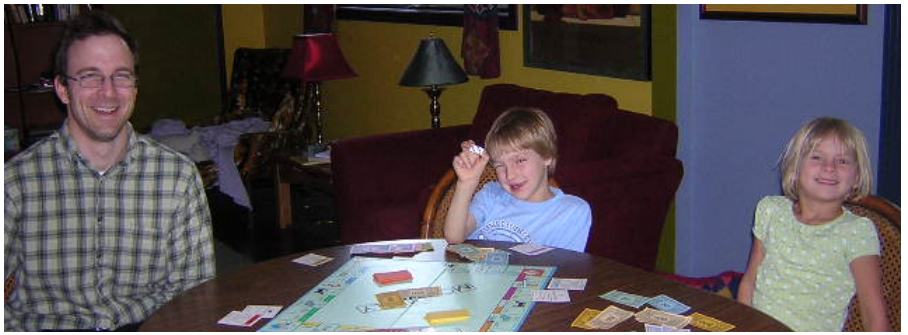
Office `Olin 1255`

Email `dodds@cs.hmc.edu`

fan of *low-tech* games
fan of *low-level* AI
Starbucks!

and not afraid of stuffed animals!



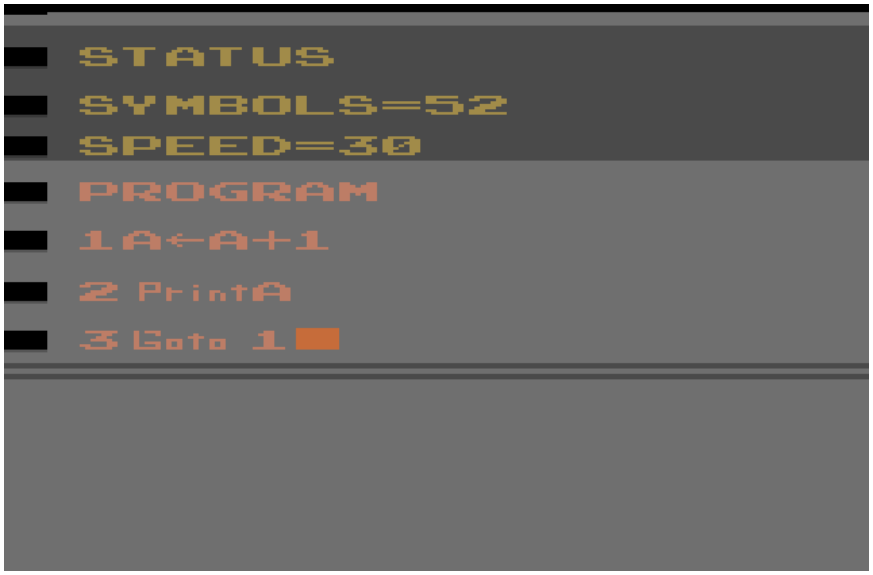


How I spent my summer vacation...

programming robots

Or, more precisely, watching many other folks programming robots!

programming *on an Atari 2600*!



Or, more precisely, watching Andrew Michaud programming the Atari 2600!

http://www.youtube.com/watch?v=wbPG8QlgruU

How I spent my summer vacation...

fan of *low-tech* games
fan of low-level AI
Starbucks!

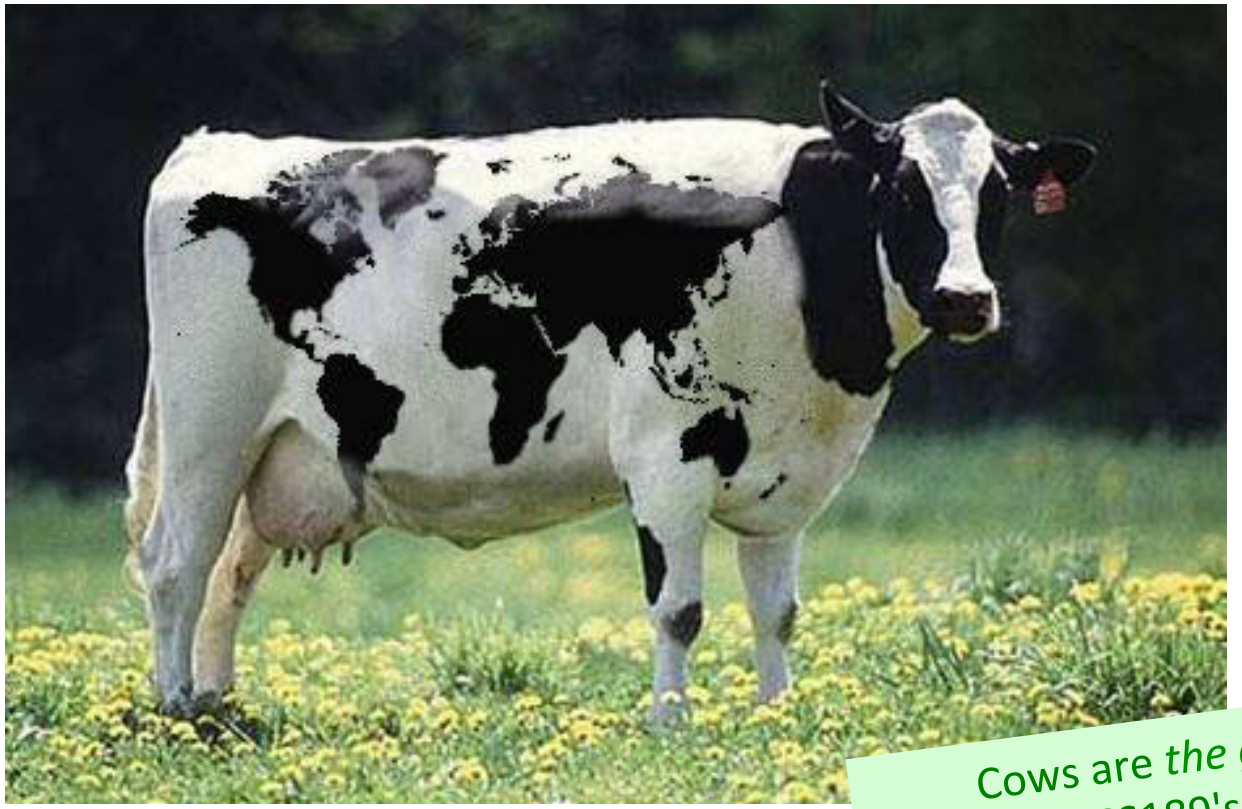
And McDonalds – anywhere!

What is this course about?

practicing algorithmic/programming skills

What is this course about?

practicing algorithmic/programming skills



Bessie!

Cows are *the* ***global*** theme of CS189's problems.

An example:

cowlpha.py or .c or .cc or .java …

Input

of upper-case letters FJ overheard, < 251

# of lower-case letters, < 251

# of valid 2-letter pairs in the Cow language

```
2 2 7
AB
ab
BA
ba
Aa
Bb
bB
```

Here are the 7 valid 2-letter pairs in the Cow language.

Output

```
7
```

The number of valid Cow-language utterances with the correct number of upper- and lower-case letters.

In this case, the seven valid Cow-words are

AabB **Abba**
abBA **BAab**
BbBb **bBAa**
bBbB

though the problem doesn't require knowing these.

What about **AaAa** ?

I suppose they'd be udderances!

What is this course about?

practicing algorithmic/programming skills

What

Algorithm analysis and insight

Program design and implementation

} optimizing ***coding*** time, as well as ***running*** time

Why

ACM programming contest

Hands-on practice with algorithms and techniques

Familiarizing with your choice of language/libraries

"reasonable"

Research/prototype programming

Technical interview questions…

Unofficial course name: CS -70

Class Organization

alternating format

discussion sessions

- problem and program analysis
- discussion of strategy and coding tips
- deciding on functional decomposition, data structures, language facilities, and algorithms to use in teams of 2-3
- short time to work on at least 1 problem

lab sessions

- more extended team problem-solving practice: coming to the problems "cold"
- these problems count for ***each*** member of the group

- sometimes new problems, other times with known ones
- ~5-6 problems given out per week…

Course Organization

Sep 11	Welcome! and DP problems ~ **5 problems**
Sep 18	Lab session ~ **5 problems**
Sep 25	Discussion session on graph problems ~ **5 problems**
Oct 2	Lab session on graph problems ~ **5 problems**
Oct 9	Discussion session on geometry problems ~ **5 problems**
Oct 16	Lab session on geometry problems ~ **5 problems**
Oct 23	**Lab & local ACM qualifying contest** ~ **6 problems**
Oct 30	Discussion session on something new!! ~ **5 problems**
Nov 6	Lab session ~ **5 problems**
Nov 10	(Sat.) **ACM Regional contest** (Riverside...)
Nov 13	Final meeting (make-up lab if needed)

≥ 42 problems total

You may submit problems
until the end of exams…

part – but only *part* – of the motivation for CS 189:

ACM programming contest

USC advanced to the finals in 2011...


78 teams...
Jackson!

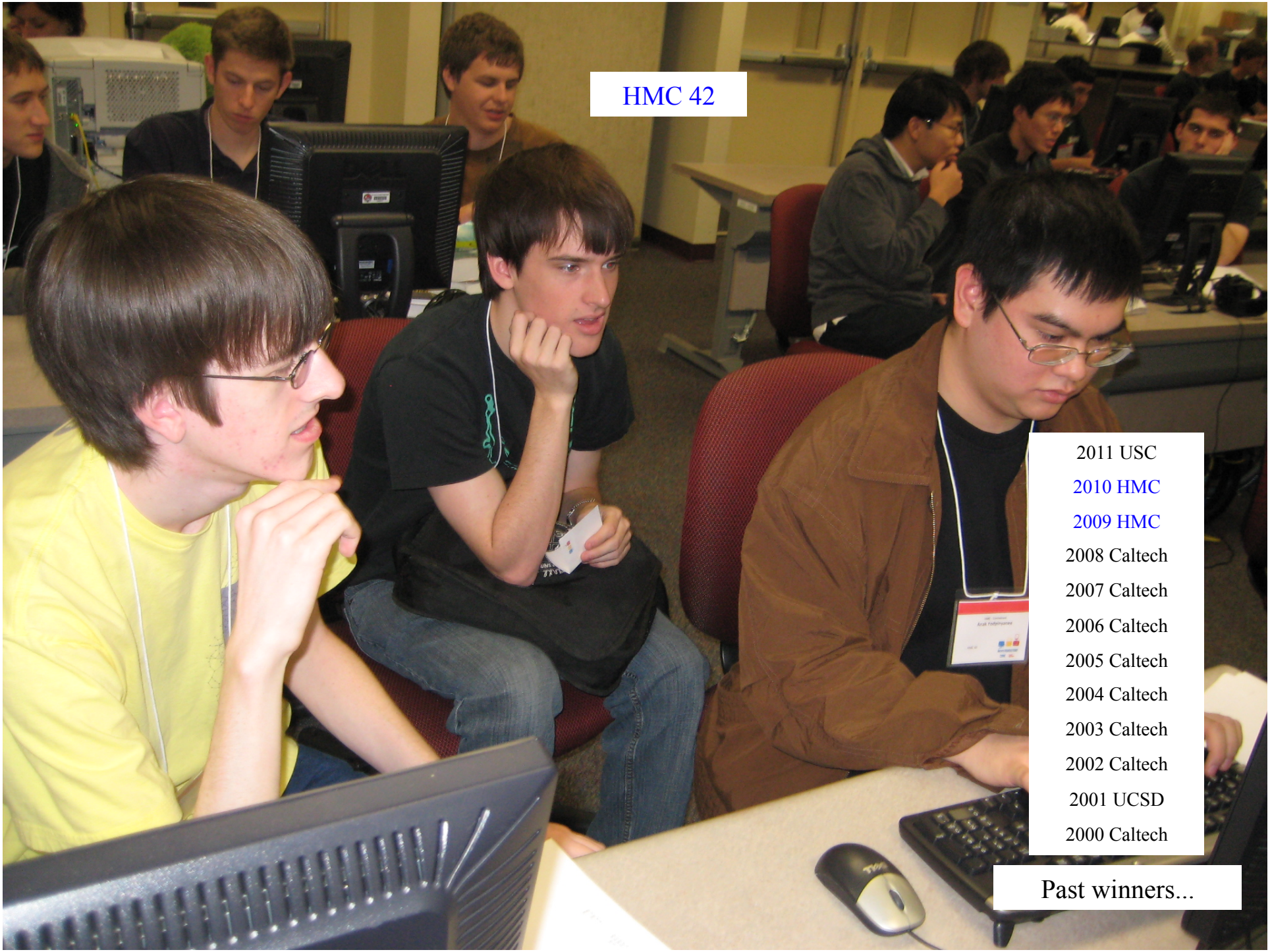

PLEASE!
No Cell Phones
No Web Surfing
No Gaming
No E-Mailing
NO ONLINE or POCKET TEXT TRANSLATORS
DELL
002 - 008
Benson!

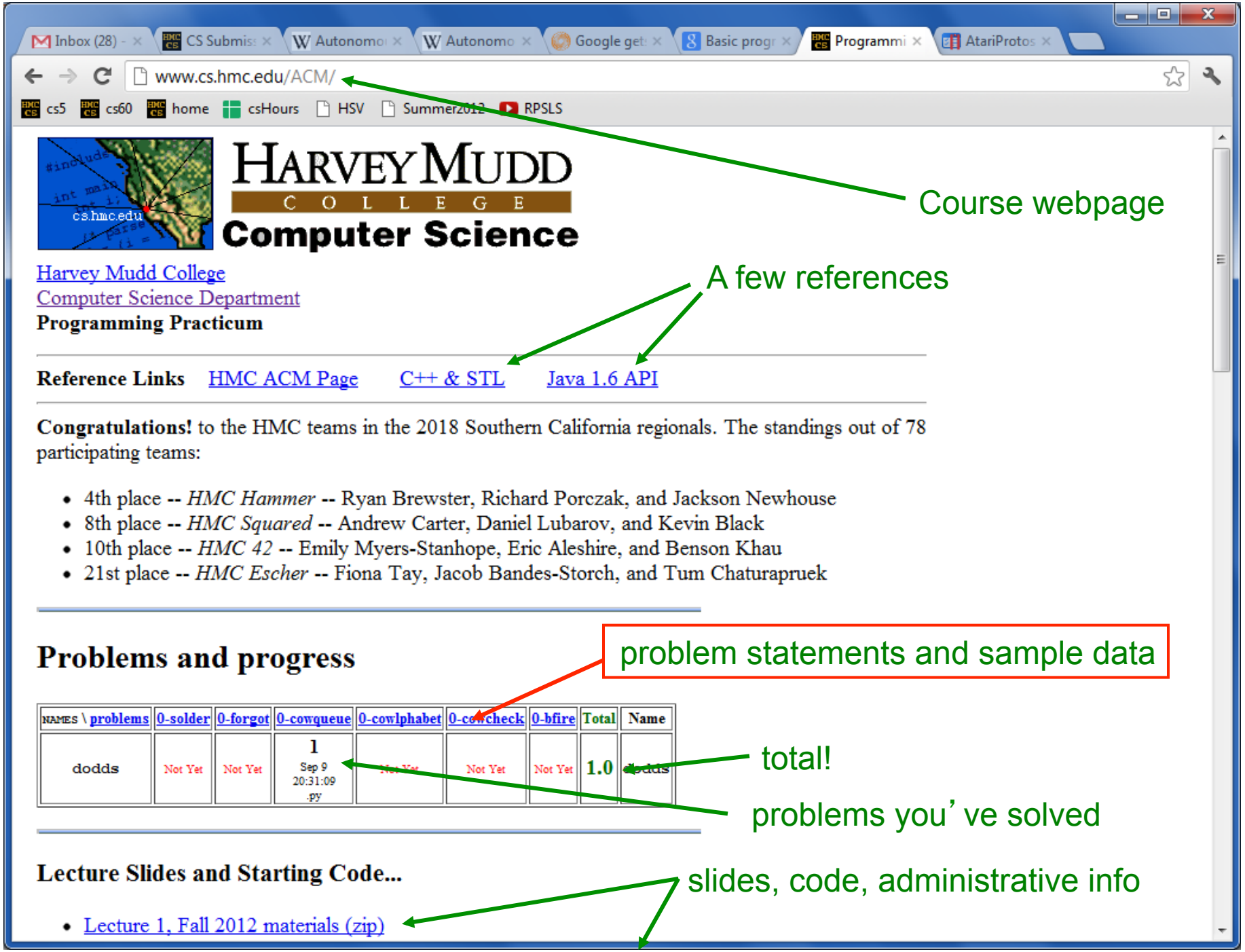
Fluxx…


active
watching!


active
watching!

active watching!


HMC 42
2011 USC
2010 HMC
2009 HMC
2008 Caltech
2007 Caltech
2006 Caltech
2005 Caltech
2004 Caltech
2003 Caltech
2002 Caltech
2001 UCSD
2000 Caltech
Past winners...


www.cs.hmc.edu/ACM/
Course webpage
HARVEY MUDD COLLEGE
Computer Science
Harvey Mudd College
Computer Science Department
Programming Practicum
A few references
Reference Links HMC ACM Page C++ & STL Java 1.6 API
Congratulations! to the HMC teams in the 2018 Southern California regionals. The standings out of 78 participating teams:
4th place -- HMC Hammer -- Ryan Brewster, Richard Porczak, and Jackson Newhouse
8th place -- HMC Squared -- Andrew Carter, Daniel Lubarov, and Kevin Black
10th place -- HMC 42 -- Emily Myers-Stanhope, Eric Aleshire, and Benson Khau
21st place -- HMC Escher -- Fiona Tay, Jacob Bandes-Storch, and Tum Chaturapruek
Problems and progress
problem statements and sample data
NAMES \ problems
0-solder
0-forgot
0-cowqueue
0-cowlphabet
0-cowcheck
0-bfire
Total
Name
dodds
Not Yet
Not Yet
1
Sep 9 20:31:09 .py
Not Yet
Not Yet
1.0
dodds
total!
problems you've solved
Lecture Slides and Starting Code...
slides, code, administrative info
Lecture 1, Fall 2012 materials (zip)

Grading

CS 189 is graded by default … (it' s possible to take it P/F, too)

though not for CS elective credit…

Coding Guidelines

- problems can be done *any time* during the semester
- discussion of algorithms always OK
- coding should be *within teams of 1-3*
- you may use any references *except* others' solutions or partial solutions…
- use **/cs/ACM/acmSubmit <file>** to submit on **knuth**
- try things out !

# Solved (out of 42)	Assessment
43+	pretty much impossible!
28-42	A
23-27	A-
20-22	B+
17-19	B
14-16	B-
9-13	C range
≤ 9	< D range or less

the reason for CS 189!

Details

the team gets credit, if you're in a team

Problems are worth 150% if

- You solve them during the 4:15 - 5:30 lab sessions
- … which extend up to about 10pm at night.

Language Choice?

Any ***reasonable*** language is OK; keep in mind that the ACM competition allows only Java, C, and C++.

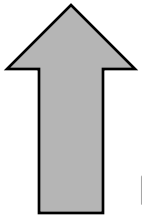
Other "standard" languages for CS189 (so far):

C#, Python, Ruby, Perl, PHP, Haskell, Lua, Clojure, Lisp

additions will also be considered…

This week's problems

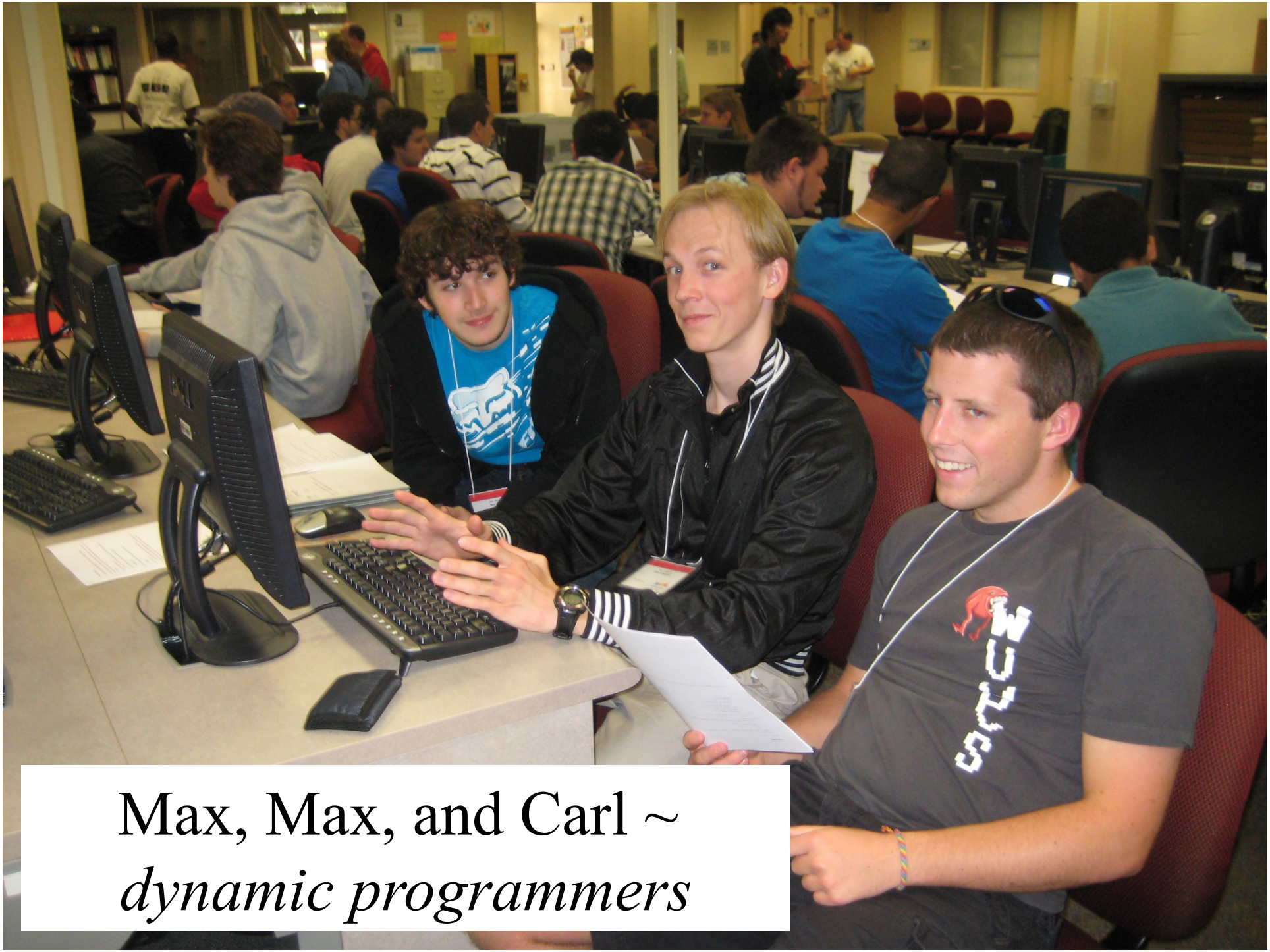
0-solder	0-forgot	0-cowqueue	0-cowlphabet	0-cowcheck	0-bfire
Not Yet	Not Yet	**1** Sep 9 20:31:09 .py	Not Yet	Not Yet	Not Yet



New to CS189? Start with this problem!

Part of the challenge is deciding ***which*** problem to tackle...

Some of this week's problems have a "dynamic programming" theme...



Max, Max, and Carl ~
dynamic programmers

Dynamic Programming

Many problems can be solved recursively...

... but with lots of ***repeated*** recursive calls!

These problems can be solved ***quickly*** with

(1) **Memoization**, or

(2) **Dynamic programming**

Idea: *just don't repeat the repeated calls!*

The **cowqueue** problem

Input

ABACB
AABC

Cow label sequence #1 (s1)

Cow label sequence #2 (s2)

Output

3

The number of the *longest common subsequence* bewteen s1 and s2.

In this case, the longest common subsequence is **ABC** or **AAB** though the problem doesn't require knowing these.

LCS problem

```
s1 = "ABACB"        Input        s2 = "AABC"
       ↑                                ↑
       i1                               i2
```

Output

LCS(i1, i2) = length of longest common subsequence of s1 up to i1 and s2 up to i2

Strategy

(1) Write a solution recursively.
(2) Then, don't make any call more than once!

LCS problem

```
s1 = "ABACB"        Input        s2 = "AABC"
       ↑                                ↑
       i1                               i2
```

length of longest common subsequence
of s1 up to i1 and s2 up to i2

LCS(i1, i2):

if s1[i1] == s2[i2]: return 1 + LCS(i1-1, i2-1)

if the same character, count it!

else: return max(LCS(i1-1, i2), LCS(i1, i2-1))

otherwise, lose both ends and take the better result

LCS code

```
s1 = "ABACB"        Input        s2 = "AABC"
        ↑                                ↑
        i1                               i2
```

cowqueue_recursive.py – /Users/zdodds/Desktop/cowqueue_recursive.py

```
import sys
sys.setrecursionlimit(100000)

def LCS( i1, i2 ):
 """ classic LCS """

 if i1 < 0 or i2 < 0: return 0

 if s1[i1] == s2[i2]:
     return 1 + LCS(i1 - 1, i2 - 1)
 else:
     return max(LCS(i1 - 1, i2), LCS(i1, i2 - 1))


if __name__ == "__main__":

 s1 = raw_input(); L1 = len(s1)
 s2 = raw_input(); L2 = len(s2)

 result = LCS( L1-1, L2-1 )

 print result
```

LCS idea

```
s1 = "ABACB"      Input      s2 = "AABC"
       ↑                           ↑
       i1                          i2
```

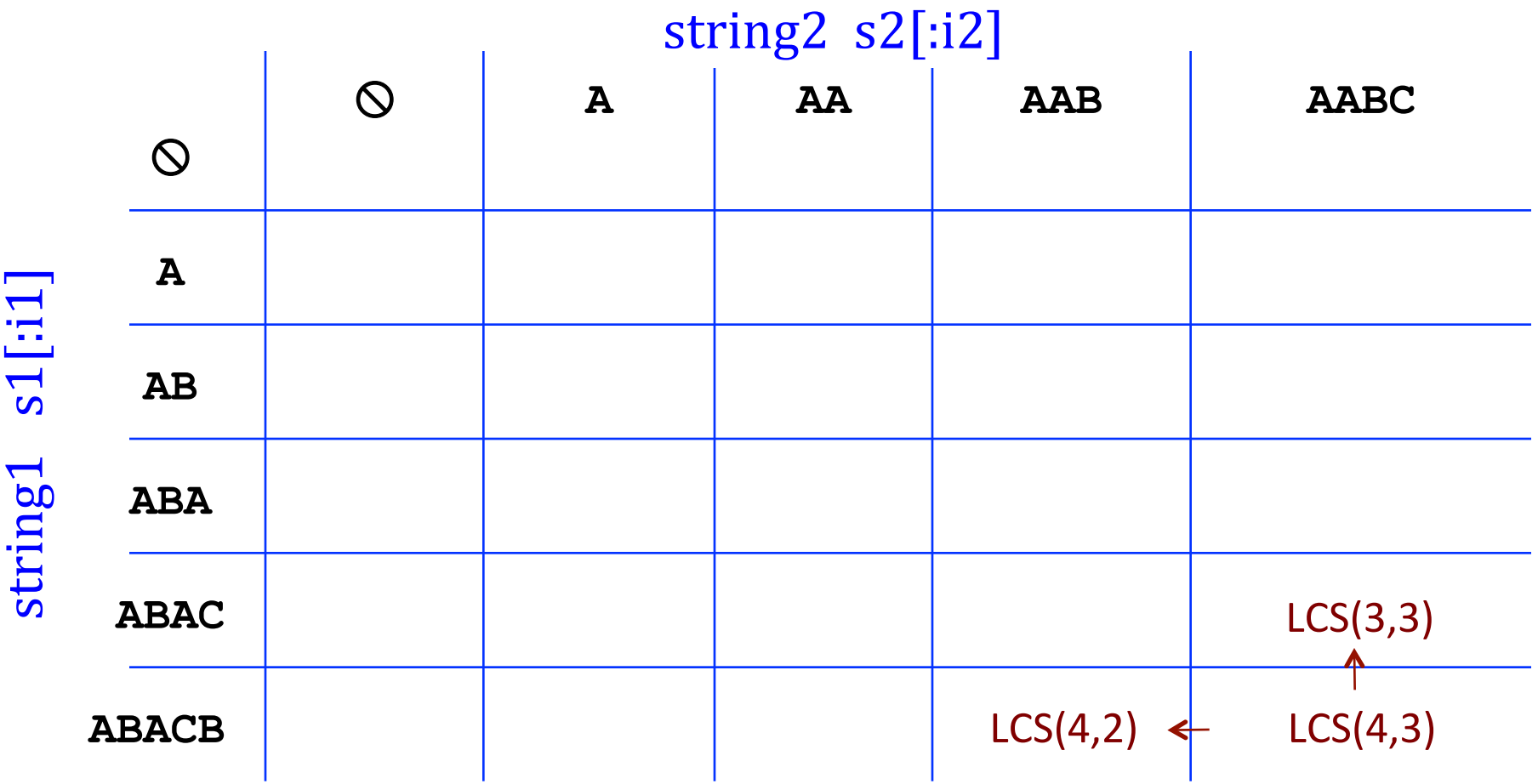
string2 s2[:i2]

string1 s1[:i1]

	⊘	A	AA	AAB	AABC
⊘					
A					
AB					
ABA					
ABAC					
ABACB					LCS(4,3)

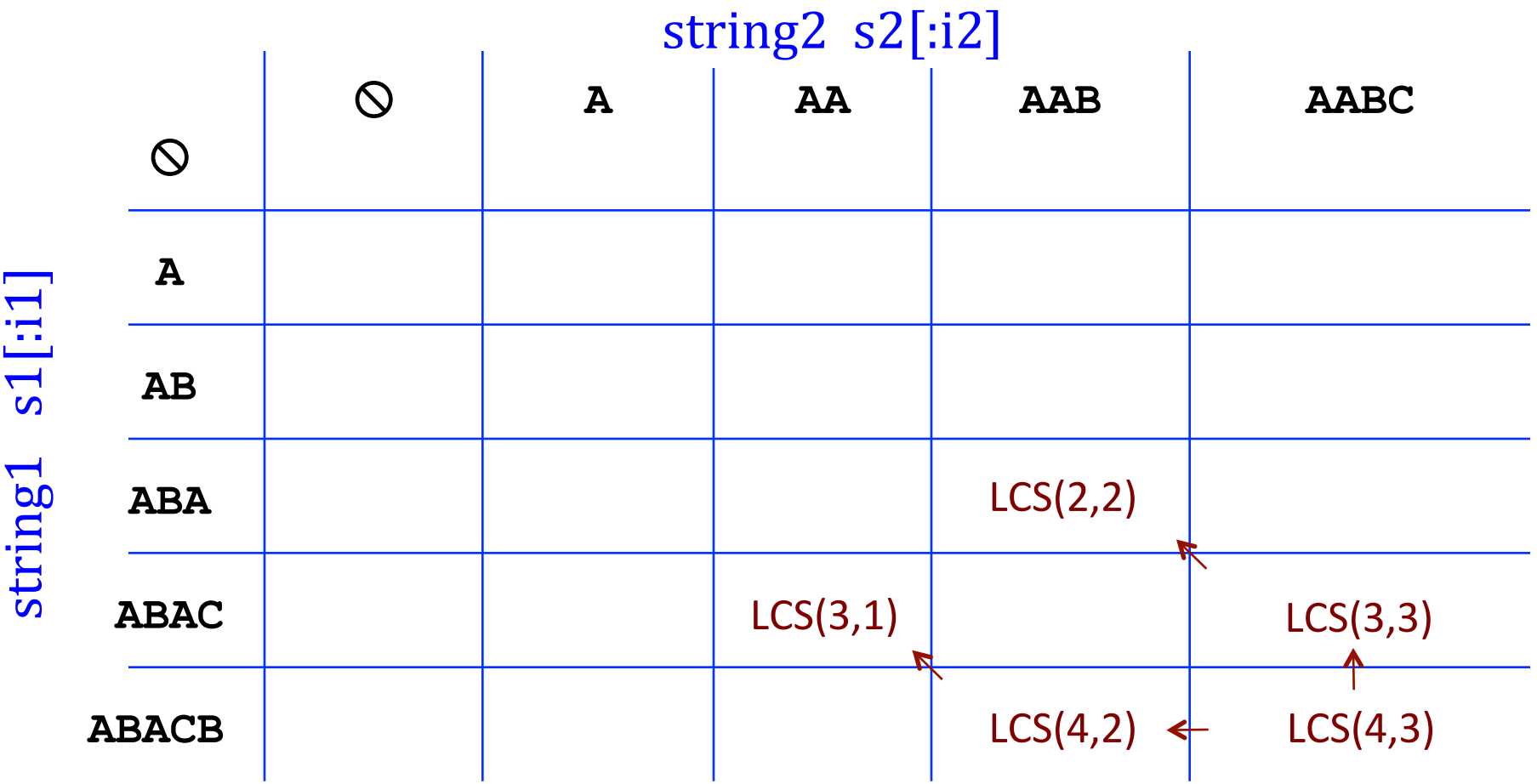
LCS idea

```
s1 = "ABACB"      Input      s2 = "AABC"
       ↑                           ↑
       i1                          i2
```



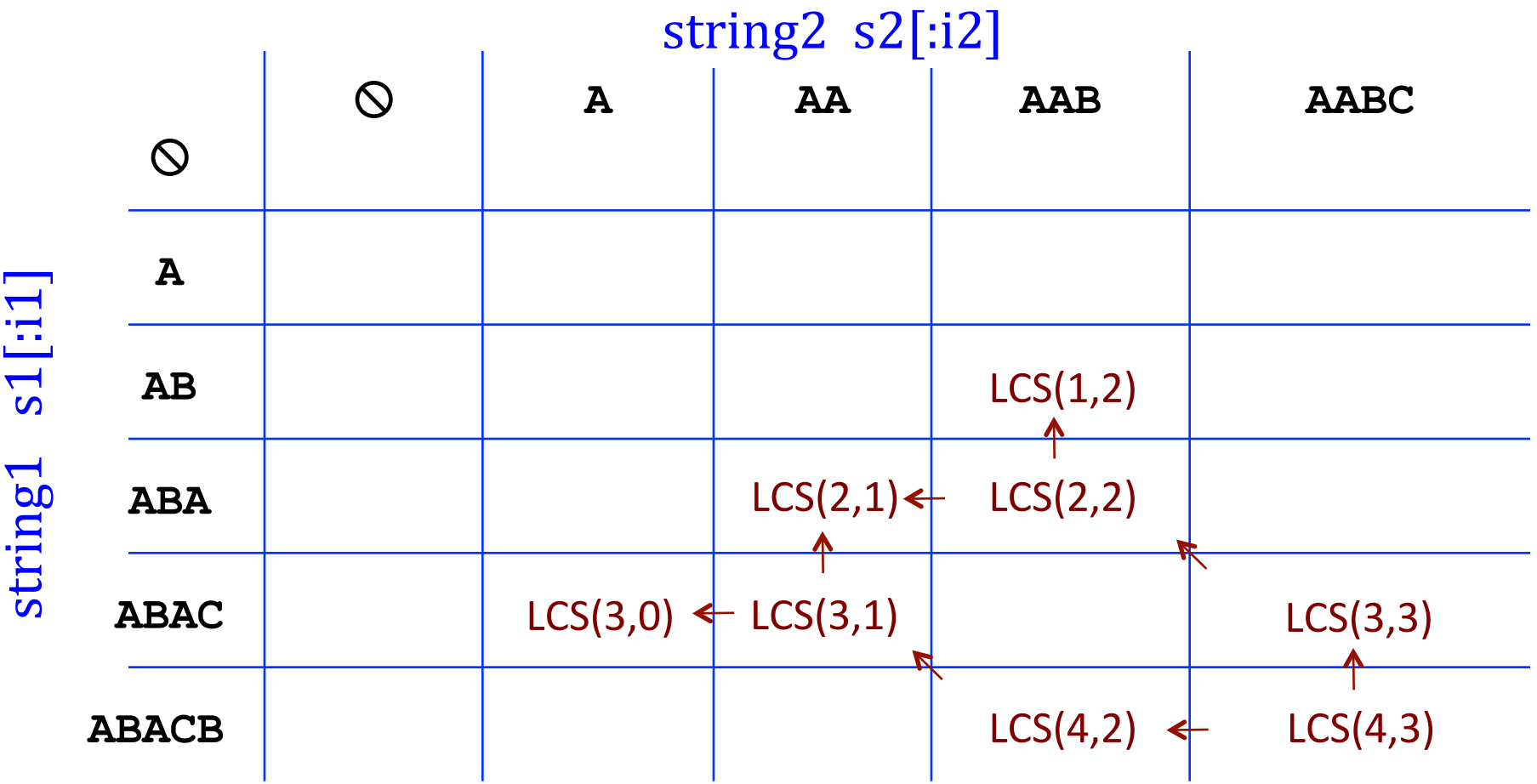
LCS idea

```
s1 = "ABACB"      Input      s2 = "AABC"
       ↑                           ↑
       i1                          i2
```



LCS idea

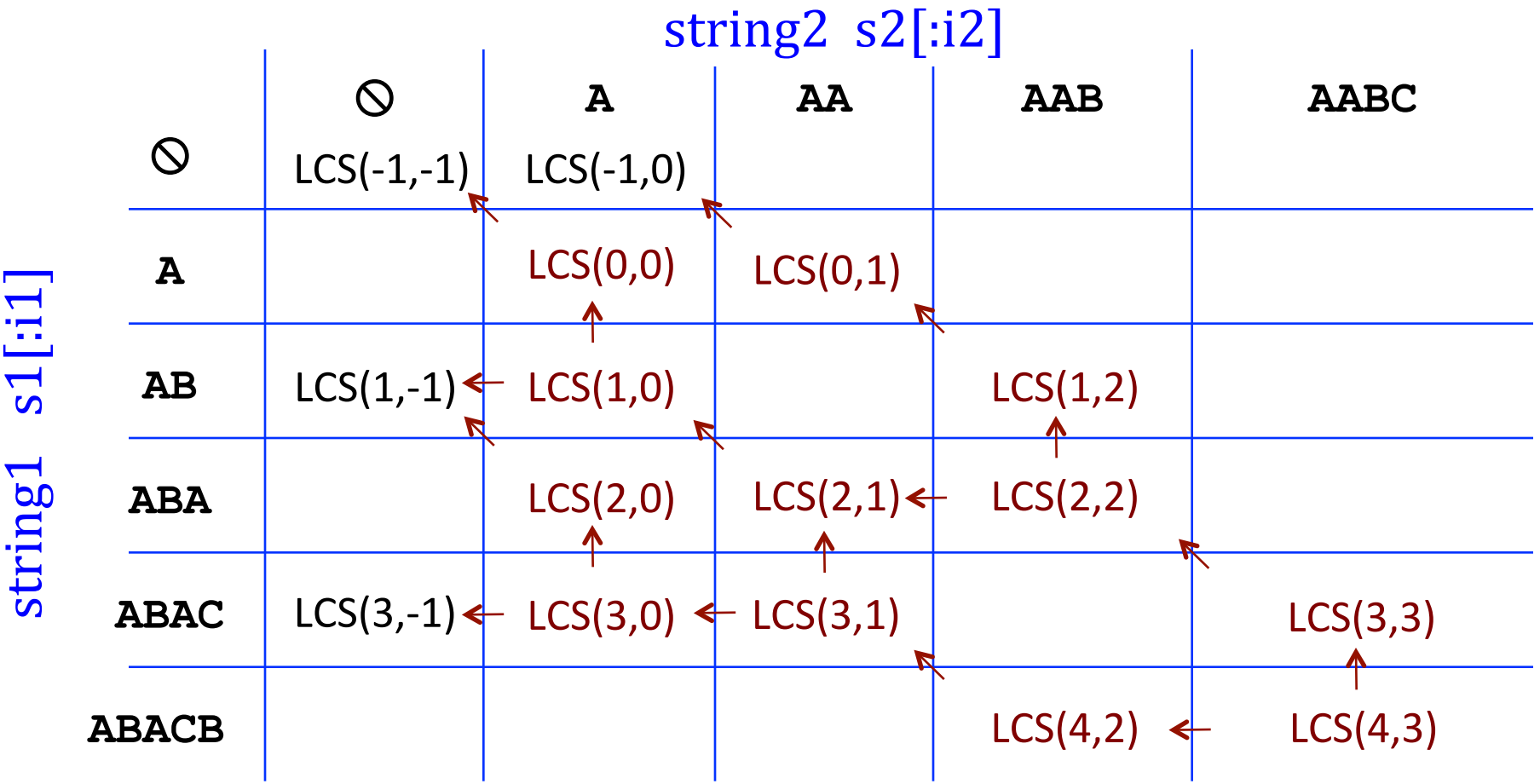
```
s1 = "ABACB"      Input      s2 = "AABC"
       ↑                           ↑
       i1                          i2
```



LCS idea

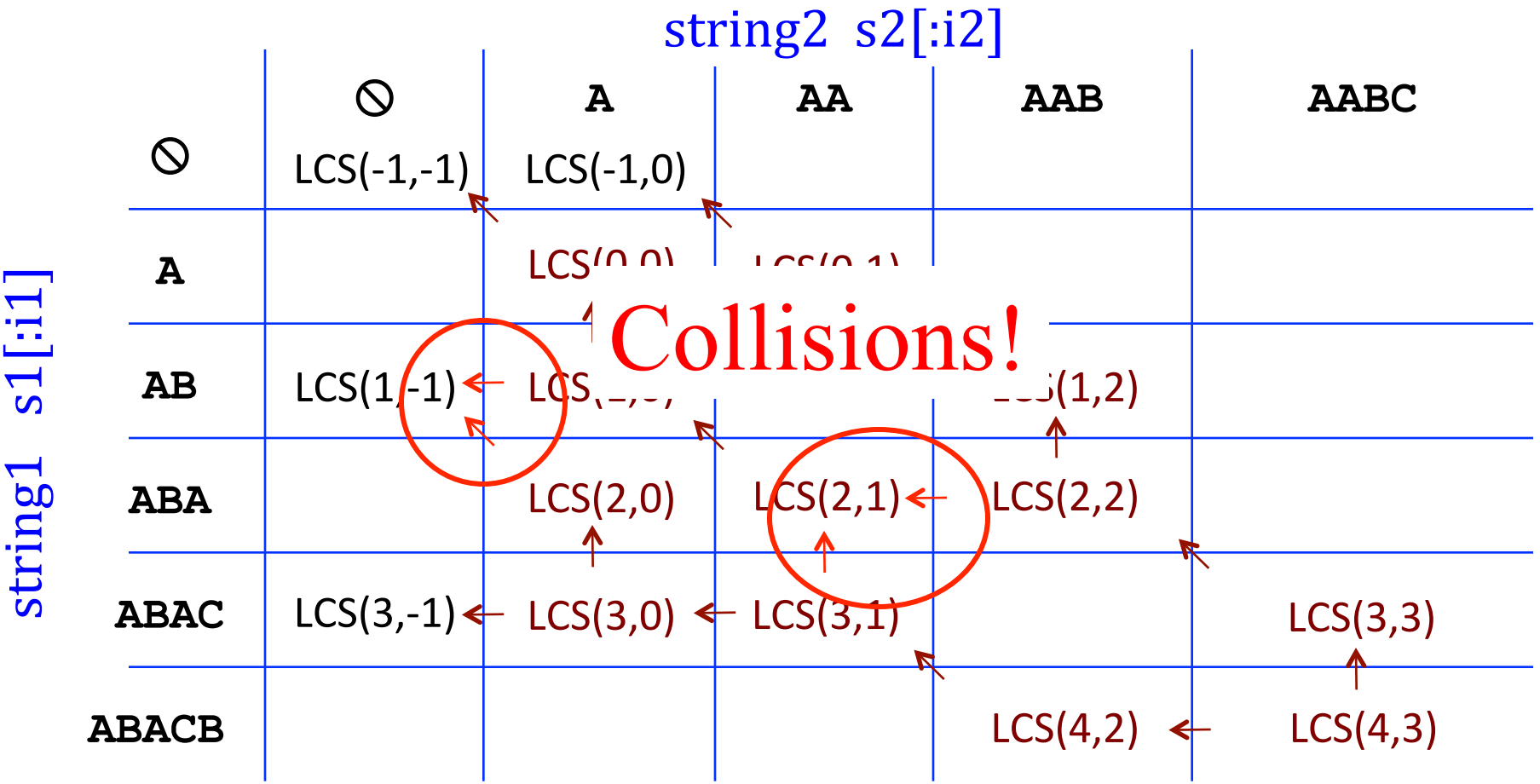
s1 = "ABACB" Input **s2 = "AABC"**

i1 i2



LCS idea

```
s1 = "ABACB"      Input      s2 = "AABC"
       ↑                           ↑
       i1                          i2
```



LCS, memoized

Put results in a dictionary.
Look up instead of recomputing.

```
# This is the "memoizing" dictionary of all distinct calls.
# Each distinct call is made only once and stored here.

D = {}

def LCS( i1, i2 ):
 """ classic LCS """

 if i1 < 0 or i2 < 0: return 0          # base cases

 if (i1,i2) in D: return D[ (i1,i2) ]   # already done!

 if s1[i1] == s2[i2]:
     result = 1 + LCS(i1-1, i2-1)
 else:
     result =  max( LCS(i1-1, i2), LCS(i1, i2-1) )

 D[ (i1,i2) ] = result                  # memo-ize it!

 return result                          # before returning


if __name__ == "__main__":

 s1 = raw_input(); L1 = len(s1)
 s2 = raw_input(); L2 = len(s2)

 result = LCS( L1-1, L2-1 )

 print result
```

Python *function decorators*

```
import sys; sys.setrecursionlimit(100000)

class memoize:
  def __init__(self, function):
    self.function = function
    self.memoized = {}

  def __call__(self, *args):
    try:
      return self.memoized[args]
    except KeyError:
      self.memoized[args] = self.function(*args)
      return self.memoized[args]
```

Python's "function decorator" syntax!

```
@memoize
def LCS( i1, i2 ):    # slow, recursive f'n here
```

LCS, DP'ed

Compute the table of results, bottom-up!

```
s1 = "ABACB"      Input      s2 = "AABC"
       ↑                            ↑
       i1                           i2
```

string2 s2[:i2]

string1 s1[:i1]	⊘	A	AA	AAB	AABC
⊘					
A					
AB					
ABA					
ABAC					
ABACB					

LCS, DP'ed

Compute the table of results, bottom-up!

```
s1 = "ABACB"        Input        s2 = "AABC"
       ↑                                ↑
       i1                               i2
```

string2 s2[:i2]

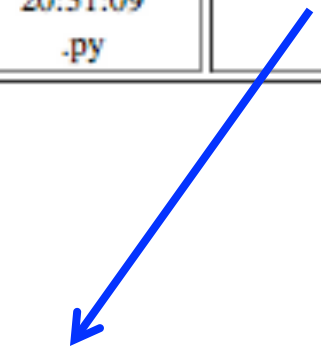
string1 s1[:i1]

	⊘	A	AA	AAB	AABC
⊘					
A					
AB					
ABA					
ABAC					
ABACB					

```
if __name__ == "__main__":

 s1 = raw_input(); L1 = len(s1)
 s2 = raw_input(); L2 = len(s2)

 DP = [ [0]*(L2+2) for i1 in range(L1+2) ]

 for i1 in range(L1):
   for i2 in range(L2):
     if s1[i1] == s2[i2]: DP[i1][i2] = 1 + DP[i1-1][i2-1]
     else: DP[i1][i2] = max( DP[i1][i2-1], DP[i1-1][i2] )

 result = DP[L1-1][L2-1]

 #for row in DP:
 #  print row

 print result
```

Try one of these!

0-solder	0-forgot	0-cowqueue	0-cowlpha	0-cowcheck	0-bfire
Not Yet	Not Yet	**1.5** Sep 9 20:31:09 .py	Not Yet	Not Yet	Not Yet



L = # of lower-case heard
U = # of upper-case heard
C = last character

Solution = DP(L, U, C)

but DP(L, U, C) =

Good LUC!

Jotto!

A word-guessing game similar to mastermind…

Sophs	JRs	SRs	Elde	Pomona
slate 1	slate 3			slate 3

Ideas for a new inter-class CS189 game !?

This term's first class to guess another's word earns 1 problem...

This term's last class to have its word guessed earns 1 problem...

Current Jotto standings…

Win!			Quine 5
Sophs	**Jrs**	**Srs**	**Others**
icily 0	icily 0	icily 1	icily 1
strep 2	strep 2	strep 2	strep 1
spork 1	spork 3	spork 0	spork 0
spend 2	spend 2	spend 2	spend 2
peeps 2	peeps 1	peeps 2	peeps 1
furls 1	furls 1	furls 0	furls 1
Ghost 2	Ghost 1	Ghost 1	Ghost 0
Tanks 2	Tanks 1	Tanks 2	Tanks 1
Gecko 2	Gecko 1	Gecko 1	Gecko 1

Another possibility...

This Spring 2011 term, you can also earn up to 9 points by building web-applications that solve some of the problems

From this first set of problems

Create a page in your CS webspace that solves the "fade to black" problem...

- HTML base page ☺
- with a CGI form of some sort...
- needs to use a CSS
- may involve spam, 42, and/or cows...
- needs a (functional) feature of your own design...
- *needs to color the strings appropriately!*

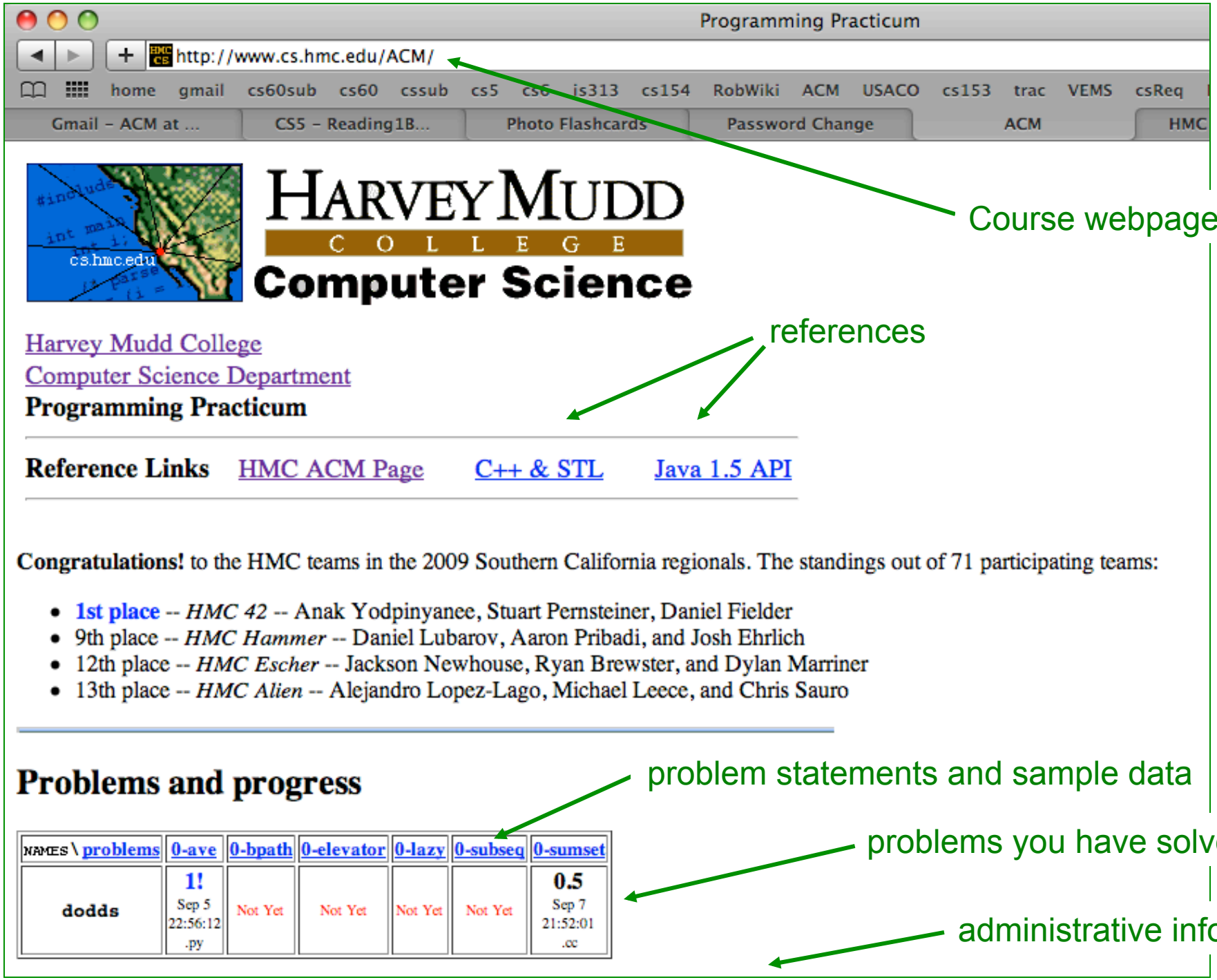
This is worth up to 2 problems.

can work in teams of 2-3, but each person should build their own. Email me your URL...

- 1st place -- *HMC 42* -- Anak Yodpinyanee, Stuart Pernsteiner, Daniel Fielder
- 9th place -- *HMC Hammer* -- Daniel Lubarov, Aaron Pribadi, and Josh Ehrlich
- 12th place -- *HMC Escher* -- Jackson Newhouse, Ryan Brewster, and Dylan Marriner
- 13th place -- *HMC Alien* -- Alejandro Lopez-Lago, Michael Leece, and Chris Sauro

Last year's ACM regional contest

Rank	Team ID	Team Name	Solved	Penalties	Score
1	acm126	HMC 42	4	2	7:54:15
2	acm105	Caltech B	4	2	9:11:21
3	acm113	SLO - Semicolons of Fury	4	5	11:07:19
4	acm104	Caltech A	4	3	13:08:38
5	acm162	USC Trojans	3	7	8:45:34
6	acm140	UC San Diego Jiandao (Scissors)	3	2	8:46:25
7	acm112	SLO - Blinkenlights	2	0	2:02:45
8	acm141	UC San Diego Yanshi (Rock)	2	3	3:54:36
9	acm129	HMC Hammer	2	2	3:56:17
10	acm161	USC Tirebiters	2	0	3:56:32
11	acm147	UCLA True Bruins	2	2	4:03:58
12	acm128	HMC Escher	2	4	4:50:26
13	acm127	HMC Alien	2	3	5:24:14
14	acm158	UNLV Ballmer Peak	2	7	6:26:44
15	acm145	UCLA IDK	2	2	7:16:13
16	acm159	USC Cardinal	2	5	7:29:32
17	acm106	Caltech C	1	0	0:23:42
18	acm114	SLO - Team Ruby Slippers	1	0	0:23:43
19	acm151	UCSB GoedelEscherBach	1	0	0:34:28
20	acm116	CPP H1N1++	1	0	0:37:17
21	acm157	UNLV - Nerd Sniping	1	0	0:40:21
22	acm154	UCSB Stroustrup's Henchmen	1	0	0:50:23
23	acm146	UCLA True Blue	1	0	0:59:24
24	acm142	UC San Diego Wenjian (Paper)	1	2	1:24:30
25	acm110	CLU NINJAS	1	0	1:47:47
26	acm152	UCSB Number Crushers	1	0	2:13:44
27	acm144	UCLA Fight! Fight! Fight!	1	1	2:33:57
28	acm111	SLO - Bad Horse's Mares	1	0	2:43:08


Programming Practicum
http://www.cs.hmc.edu/ACM/
home gmail cs60sub cs60 cssub cs5 cs6 js313 cs154 RobWiki ACM USACO cs153 trac VEMS csReq
Gmail - ACM at ... CS5 - Reading1B... Photo Flashcards Password Change ACM HMC
cs.hmc.edu
HARVEY MUDD
COLLEGE
Computer Science
Harvey Mudd College
Computer Science Department
Programming Practicum
Reference Links HMC ACM Page C++ & STL Java 1.5 API
Congratulations! to the HMC teams in the 2009 Southern California regionals. The standings out of 71 participating teams:
1st place -- HMC 42 -- Anak Yodpinyanee, Stuart Pernsteiner, Daniel Fielder
9th place -- HMC Hammer -- Daniel Lubarov, Aaron Pribadi, and Josh Ehrlich
12th place -- HMC Escher -- Jackson Newhouse, Ryan Brewster, and Dylan Marriner
13th place -- HMC Alien -- Alejandro Lopez-Lago, Michael Leece, and Chris Sauro
Problems and progress
NAMES \ problems
0-ave
0-bpath
0-elevator
0-lazy
0-subseq
0-sumset
dodds
1!
Sep 5 22:56:12 .py
Not Yet
Not Yet
Not Yet
Not Yet
0.5
Sep 7 21:52:01 .cc
Course webpage
references
problem statements and sample data
problems you have solved
administrative info

Grading

CS 189 is graded individually... (it's possible to take it P/F, too)

though not for CS elective credit…

Coding Guidelines

- problems can be done *any time* during the semester
- discussion of algorithms always OK
- coding should be *within teams*
- you may use any references *except* an existing solution or partial solution…
- one person should take the lead on each problem
- use **/cs/ACM/acmSubmit <file>** to submit on **knuth**
- try things out !

the reason for CS 189!

# Solved (out of 36)	Assessment
33+	crazy
29-32	A
25-28	A-
22-24	B+
18-21	B
15-17	B-
10-14	C range
≤ 9	< D range or less

Problem *credits*

Problems are worth double if

- You solve them during the 4:15 - 5:30 lab sessions

the team gets credit, if in a team

- It's one of the "extra difficult" problems, which may be determined as we go…

Language Choice?

- Any ***standard*** language is OK -- but do keep in mind that the competition allows only Java, C, and C++ .

Other "standard" languages (so far): C#, Python, Ruby, Perl, PHP, Haskell, Lua, Clojure, Lisp

reasonable alternatives will also be considered…

A prior term's summary: *languages*

0-archives	0-books	0-pebbles	0-roman	1-dna	1-fair	1-scrambled	1-towers	2-debt	2-dolls	2-teamwork	2-telephone	3-alienwar	3-domain	3-moat	3-trompin
17 (+2)	**8**	**4**	**16** (+2)	**1** (+1)	**6** (+1)	**20** (+12) (+2)	**3** (+1)	**1**	**1**	**2**	**11** (+1)	**1** (+1)	**17** (+9) (+2)	**4** (+2)	**1**

4-expressions	4-hull	4-jungle	4-pipeline	5-meteor	5-points	5-role	5-tragedy	6-frogger	6-lakes	6-marbles	6-mcommand	6-rubik
2	**2**	**8**	**3**	**8** (+2)	**2**	**13** (+10)	**14** (+9)	**1**	**21** (+16) (+4)	**1**	**1**	**15** (+14)

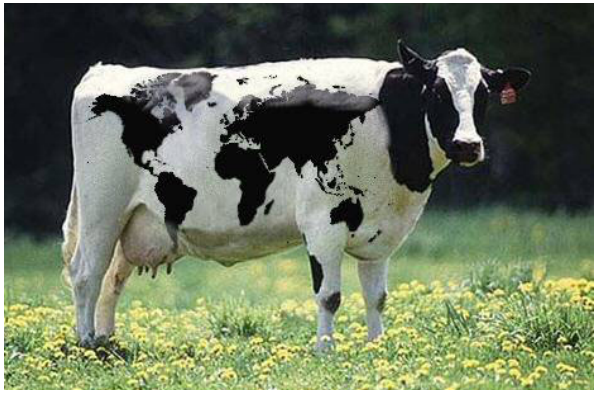
number of 2x scores

number of 4x scores

Tallies per problem and per language…

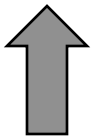
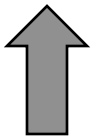
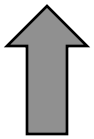
python	82	d	8	lua	2	1 each	sql, cobol, basic, x86 asm, pascal, mathematica, sh, latex
java	60	ruby	6	awk	2		
C++	40	scheme	3	js	2		

This week's problems



Cows are *the* ***global*** theme of CS189's problems.

0-ave	0-bpath	0-elevator	0-lazy	0-subseq	0-sumset
1! Sep 5 22:56:12 .py	Not Yet	Not Yet	Not Yet	Not Yet	**0.5** Sep 7 21:52:01 .cc

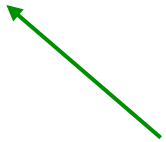




These 3 problems are worth 0.5 points, but 1 point if solved "in lab."

The **ave** problem

Input

MIX



Farmer Alvarado's roman numeral

Output

```
Ave, Mvnde!
Ave, Mvnde!
...
Ave, Mvnde!
Ave, Mvnde!
```

total of 1009 of these

That number of "Hello, World"s – in Latin.

If it's not a valid roman numeral, output

nocens numerus

i.e., *bad number*

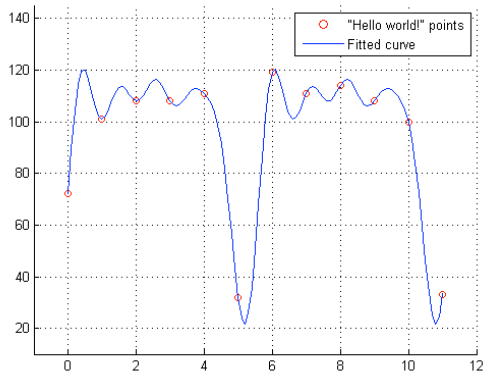
http://www.roesler-ac.de/wolfram/hello.htm

The *Hello, world!* collection

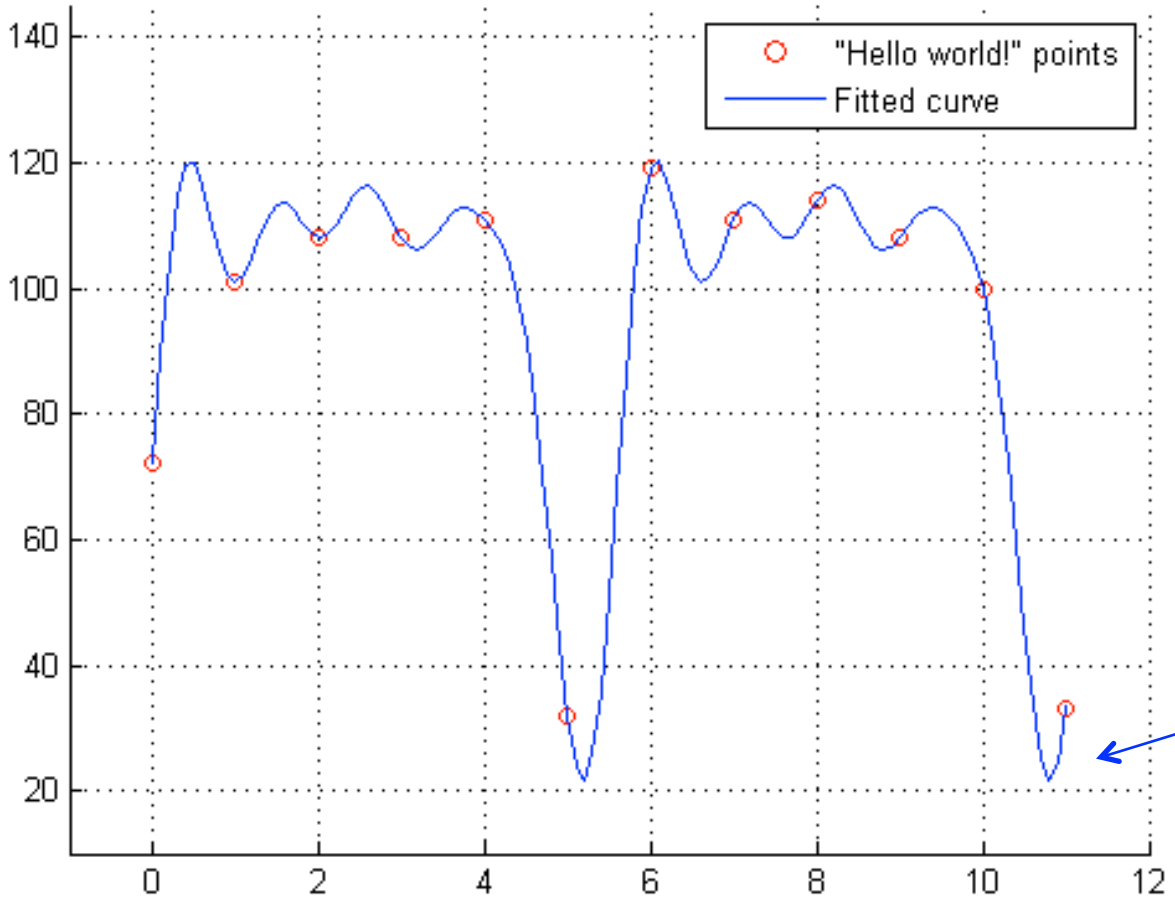
"Hello, World" of the day!

sine & cosine functions!

```
from math import *

def f(x):
    return int(round(96.75 + -21.98*cos(x*1.118)\
                           +  13.29*sin(x*1.118)\
                           + -8.387*cos(2*x*1.118)\
                           +  17.94*sin(2*x*1.118)\
                           +  1.265*cos(3*x*1.118)\
                           +  16.58*sin(3*x*1.118)\
                           +  3.988*cos(4*x*1.118)\
                           +  8.463*sin(4*x*1.118)\
                           +  .3583*cos(5*x*1.118)\
                           +  5.878*sin(5*x*1.118)))

print "".join([chr(f(x)) for x in range(12)])
```



"Hello, World" of the day!



Here is "Hello World"
How does it work?

http://www.korokithakis.net/node/89

Dynamic Programming

When a seemingly intractable problem has lots of repeated substructure, go DP!

0-ave	0-bpath	0-elevator	0-lazy	0-subseq	0-sumset
1! Sep 5 22:56:12 .py	Not Yet	Not Yet	Not Yet	Not Yet	**0.5** Sep 7 21:52:01 .cc

Build a table of partial results.

Replace computation with table look-up when possible

Dynamic programming?

Action!

Dynamic programming!

The **subseq** problem

Input

Cow label sequence #1 (s1)

```
THEQUICKBROWNCOW
JUMPEDOVERADOG
```

Cow label sequence #2 (s2)

Output

3

The number of the *longest common subsequence* bewteen s1 and s2.

what is it?

EOO

The **subseq** problem

Input

s1 = "THEQUICKBROWNCOW" **s2 = "JUMPEDOVERADOG"**

Output

LCS(i1, i2) = length of longest common subsequence of s1 up to i1 and s2 up to i2

overlapping subproblems

Strategy

(1) Write your solution recursively, in terms of these.
(2) Make sure you don't call any subproblem more than once!

The **subseq** problem

s1 = "THEQUICKBROWNCOW" **s2 = "JUMPEDOVERADOG"**

Output **LCS(i1, i2)** = length of longest common subsequence of s1 up to i1 and s2 up to i2

LCS(i1, i2) =

Recursive LCS

```
import sys
sys.setrecursionlimit(100000)

def LCS( i1, i2 ):
  """ classic LCS: s1 up to i1 vs. s2 up to i2, inclusive """

  if i1 < 0 or i2 < 0: return 0

  result = Compute LCS recursively!

  return result


if __name__ == "__main__":
  s1 = raw_input()
  s2 = raw_input()

  result = LCS( len(s1)-1, len(s2)-1 )

  print result
```

Avoiding repeated calls...

```
import sys
sys.setrecursionlimit(100000)

D = {}

def LCS( i1, i2 ):
  """ classic LCS: s1 up to i1 vs. s2 up to i2, inclusive """
  global D
  if i1 < 0 or i2 < 0: return 0

  if (i1,i2) in D: return D[(i1,i2)]  # memoized!!

  result = Compute LCS recursively!

  D[(i1,i2)] = result   # memoized!!

  return result


if __name__ == "__main__":
  s1 = raw_input()
  s2 = raw_input()

  result = LCS( len(s1)-1, len(s2)-1 )

  print result
```

Memoize

remember old calls &
don't recompute them

Avoiding repeated calls...

Memoize

remember old calls &
don't recompute them

Dynamic Programming

make each call once and
store it in a table

```
import sys
sys.setrecursionlimit(100000)

class memoize:
  def __init__(self, function):
    self.function = function
    self.memoized = {}

  def __call__(self, *args):
    try:
      return self.memoized[args]
    except KeyError:
      self.memoized[args] = self.function(*args)
      return self.memoized[args]

@memoize
def LCS( i1, i2 ):      # definition here
```

Python's "function decorator" syntax!

Avoiding repeated calls...

Memoize

remember old calls &
don't recompute them

Dynamic Programming

make each call once and
store it in a table

```
if __name__ == "__main__":
  s1 = raw_input()
  s2 = raw_input()

  D = {}  # empty table

  for i1 in range(len(s1)):
    for i2 in range(len(s2)):
      D[(i1,i2)] = LCS implementation!

  print D[(len(s1)-1, len(s2)-1)]
```

Jotto!

A word-guessing game similar to mastermind…

Sophs	Jrs	Srs	Profs
pluot 1	**pluot 2**	**pluot 1**	**pluot 2**
squid 2	**squid 1**	**squid 0**	**squid 1**

This term's winning team earns 1 problem (per person)...

Try these...

0-ave	0-bpath	0-elevator	0-lazy	0-subseq	0-sumset
1! Sep 5 22:56:12 .py	Not Yet	Not Yet	Not Yet	Not Yet	**0.5** Sep 7 21:52:01 .cc

Fall '09 Jotto: *tied for the longest game yet…*

Sophs	Jrs	Srs	Others
icily 0	icily 0	icily 1	icily 1
strep 2	strep 2	strep 2	strep 1
spork 1	spork 3	spork 0	spork 0
spend 2	spend 2	spend 2	spend 2
peeps 2	peeps 1	peeps 2	peeps 1
furls 1	furls 1	furls 0	furls 1
Ghost 2	Ghost 1	Ghost 1	Ghost 0
Tanks 2	Tanks 1	Tanks 2	Tanks 1
Gecko 2	Gecko 1	Gecko 1	Gecko 1

Dynamic programming?

practice contest…

A prior term's summary: *languages*

0-archives	0-books	0-pebbles	0-roman	1-dna	1-fair	1-scrambled	1-towers	2-debt	2-dolls	2-teamwork	2-telephone	3-alienwar	3-domain	3-moat	3-trompin
17 (+2)	**8**	**4**	**16** (+2)	**1** (+1)	**6** (+1)	**20** (+12) (+2)	**3** (+1)	**1**	**1**	**2**	**11** (+1)	**1** (+1)	**17** (+9) (+2)	**4** (+2)	**1**

4-expressions	4-hull	4-jungle	4-pipeline	5-meteor	5-points	5-role	5-tragedy	6-frogger	6-lakes	6-marbles	6-mcommand	6-rubik
2	**2**	**8**	**3**	**8** (+2)	**2**	**13** (+10)	**14** (+9)	**1**	**21** (+16) (+4)	**1**	**1**	**15** (+14)

number of 2x scores

number of 4x scores

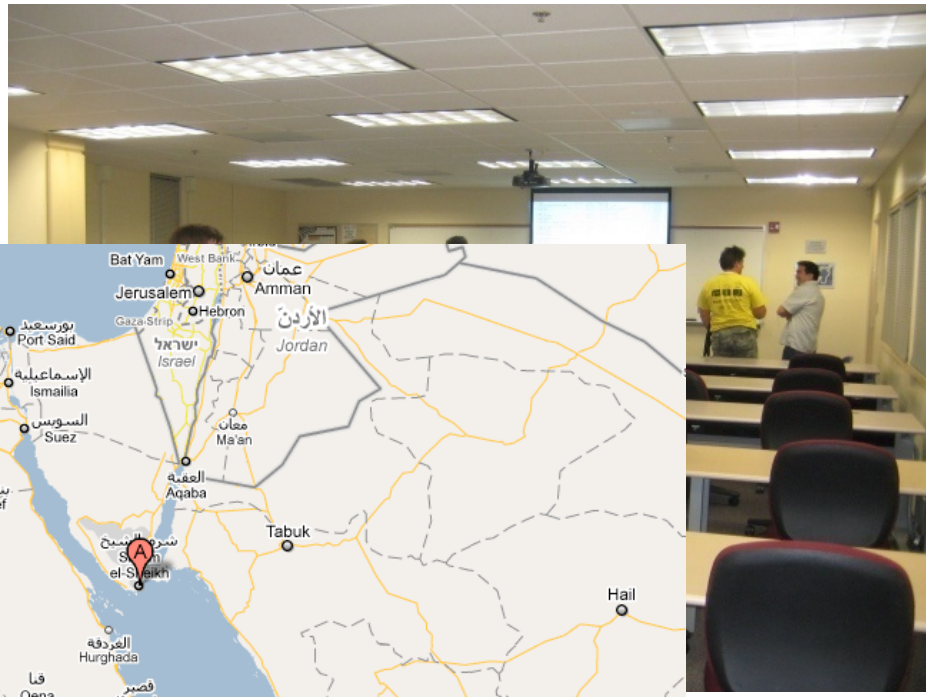
Tallies per problem and per language…

python	82	d	8	lua	2	1 each	sql, cobol, basic, x86 asm, pascal, mathematica, sh, latex
java	60	ruby	6	awk	2		
C++	40	scheme	3	js	2		

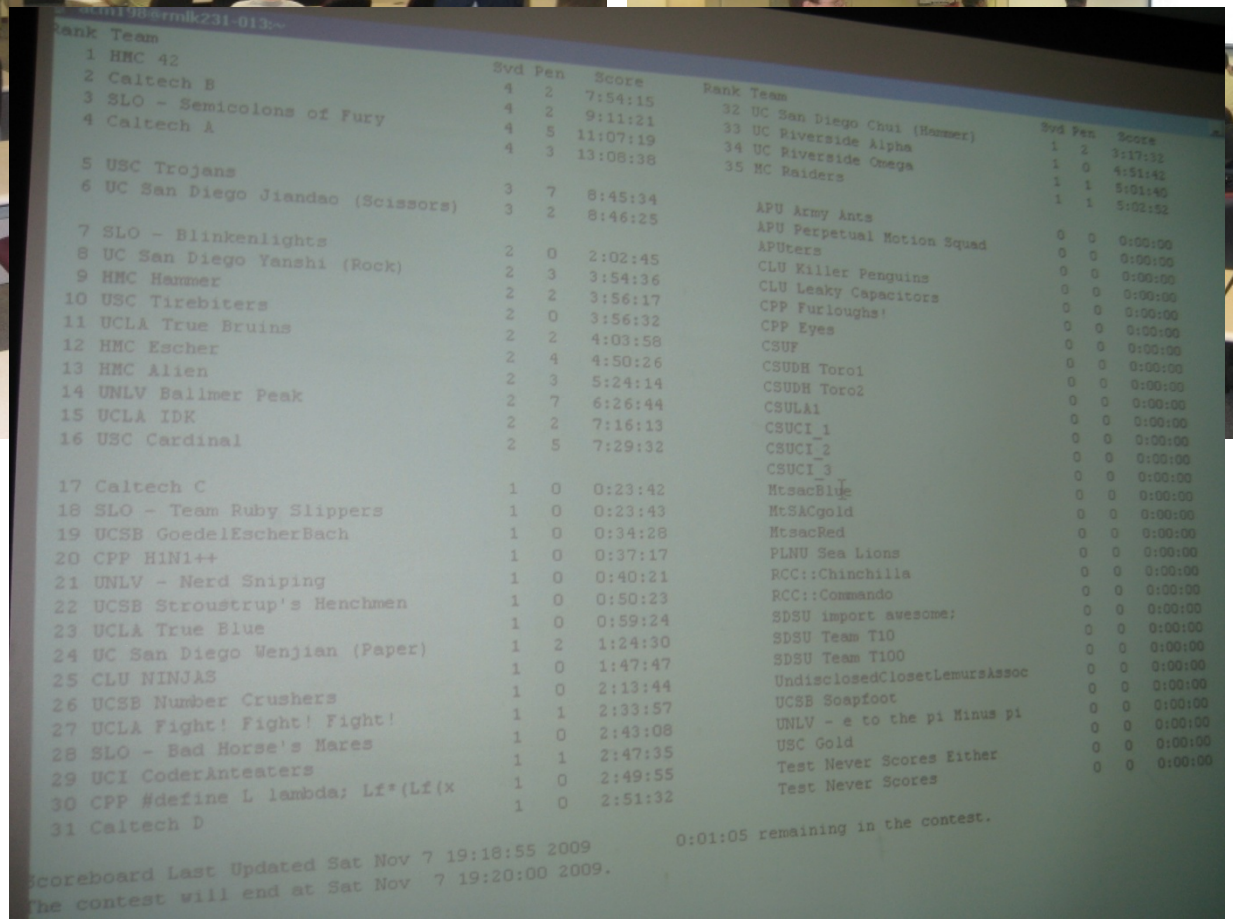
Dynamic programming?

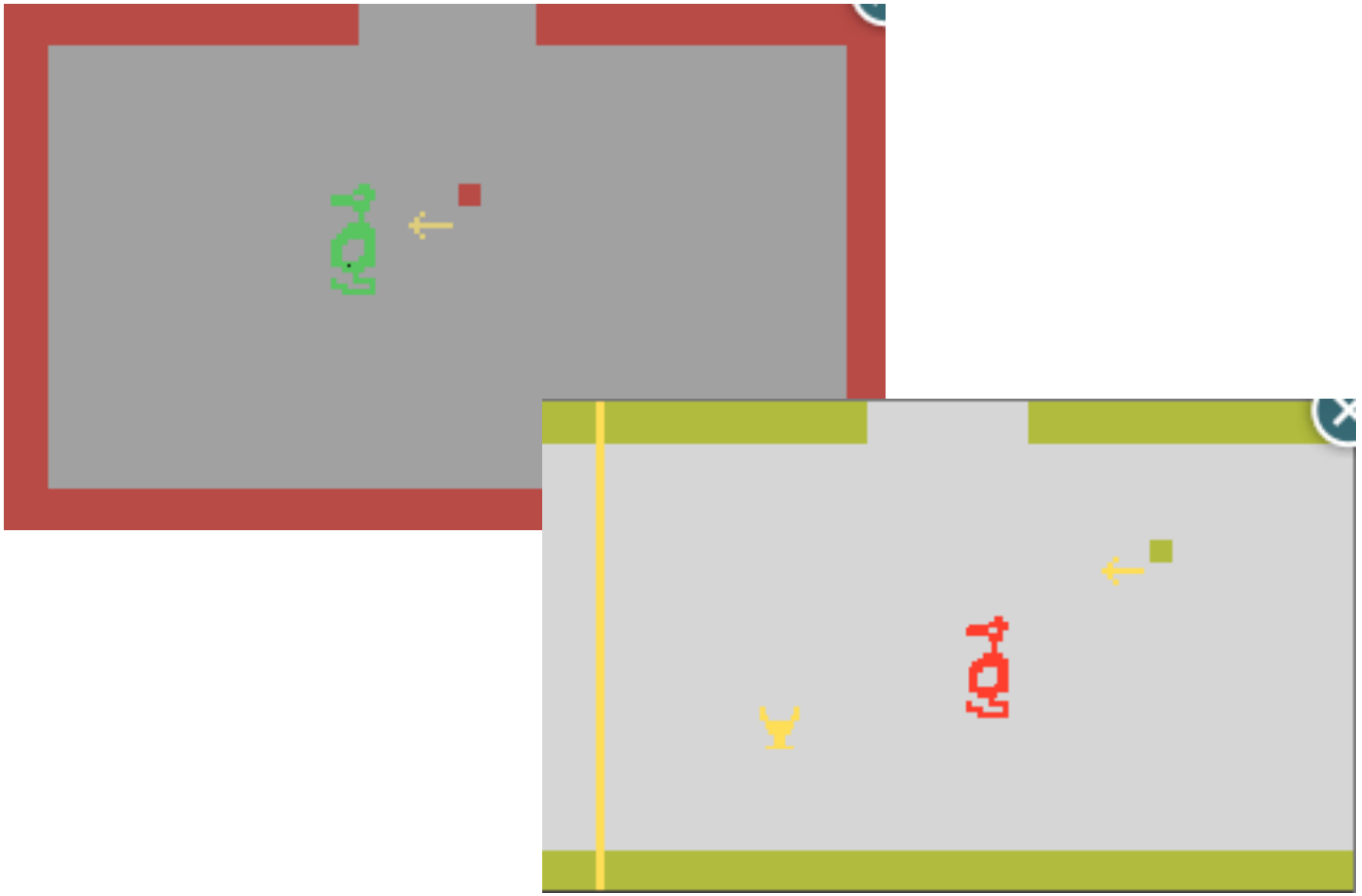
Action!

Excitement from the judges' room



Excitement from the judges' room





Atari 2600 *Adventure!*

CMC team

poker...

Content Dated 14-Nov-2010 — Updated undefined undefined

2010-11 Southern California Regional Programming Contest Final Standings

Rank	Team ID	Team Name	Solved	Penalties	Score
1	acm103	HMC 42	7	4	11:07:48
2	acm113	UCSD Papyrus	7	3	13:19:30
3	acm104	HMC Squared	6	3	11:26:40
4	acm117	Caltech i	6	5	13:07:51
5	acm118	Caltech j	6	0	13:19:42
6	acm143	UCLA Stack Overflow	6	4	13:45:35
7	acm119	Caltech k	5	2	10:05:30
8	acm165	USC Trojans	4	1	4:55:02
9	acm138	CalPoly SLO The Dark Byte	4	1	6:38:15
10	acm167	USC Gold	4	2	6:44:28
11	acm136	CalPoly SLO Team Mohawk	4	4	8:37:14
12	acm166	USC Cardinal	4	3	9:23:32
13	acm133	CalPoly Pomona Cry Havoc	4	5	9:44:25
14	acm115	UCSD Scissors	4	3	9:59:09
15	acm130	SBCC++	4	1	10:11:35
16	acm162	CSULA 1	4	2	10:49:13
17	acm105	HMC Hammer	4	2	10:50:24
18	acm137	CalPoly SLO Team Gold Star	3	0	2:09:46
19	acm114	UCSD Rock	3	0	2:32:30
20	acm150	UCR Leonardo	3	0	3:14:08

I approve of this name!

Both of these teams advanced to the finals in 2011...

Problem 7 – Daniel?!

Problem 6 – Stuart Only!

Problem 7
Zombie Blast! (continued)

Sample Input

```
M..Z
..ZZ
B..Z

.ZZ.M
Z.Z..
.Z.ZZ
Z.Z.Z

...M.....
.........
.M.......
.........
.........
.........
.........
.........
...Z.....

B
```

```
3.16
5.00
6.32
0.00
```

Sample Input

```
---U---[!X3]---[X1]--U-----(Y2)-->
   L---[X3]----[!X1]-L
-----[X3]---[X1]-------U--(R00)-U-->
                          L--(Y9)--L
          U---[!X0]--[!X2]---[R00]--U
----U-----L----[X0]---[X2]---[R00]--L------U----------(Y1)-->
    L-U--------[X0]--[!X2]--[!R00]-----U---L
      L-------[!X0]---[X2]--[!R00]-----L
--U----[X0]----[X2]-----------------U-------------(Y0)-->
  L-----[R00]-------U-----[X0]----U--L
                      L-----[X2]----L
---[X9]------U----[Y0]----U---(Y8)--->
              L----[Y8]----L
```