**University of California, Berkeley – College of Engineering, EECS Dept**

Spring 2011 Instructor: Dan Garcia 2011-05-11

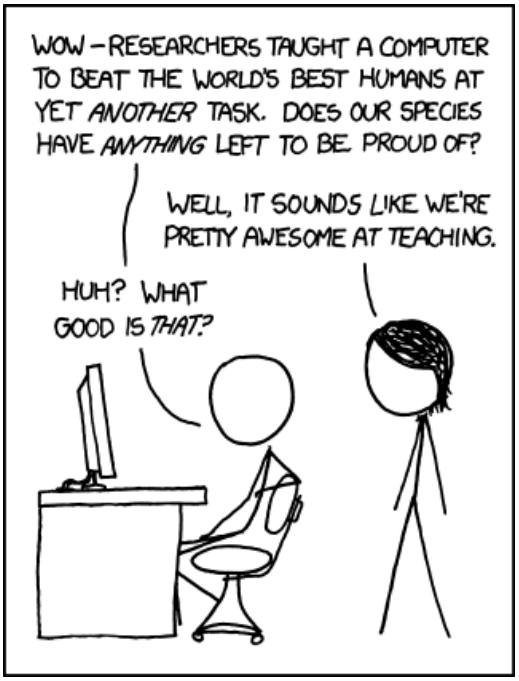
# CS10 Paper Final Exam

| | |
|---|---|
| *Last Name* | |
| *First Name* | |
| *Student ID Number* | |
| *cs10- Login First Letter* | **a b c d e f g h i j k l m** |
| *cs10- Login Last Letter* | **a b c d e f g h i j k l m**<br>**n o p q r s t u v w x y z** |
| *The name of your LAB TA (please circle)* | **Glenn Luke Navin** |
| *Name of the person to your Left* | |
| *Name of the person to your Right* | |
| *All my work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CS10 who have not taken it yet.* ***(please sign)*** | |

## Instructions

- This booklet contains 4 double-sided pages including this cover page. Put all answers on these pages; don't hand in any stray pieces of paper.
- Please turn off all pagers, cell phones and beepers. Remove all hats and headphones.
- You have 180 minutes to complete this exam. This final is closed book, no computers, no PDAs, no cell phones, no calculators, but you are allowed three double-sided pages of notes. There may be partial credit for incomplete answers; write as much of the solution as you can. When we provide a blank, please fit your answer within the space provided.

| Q | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | Online | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Pts | 4 | 4 | 8 | 4 | 4 | 4 | 4 | 4 | 4 | 5 | 10 | 10 | 15 | 80 |
| # | | | | | | | | | | | | | | |



# Short-answer Questions (this page only)

**Question 1** : HCI researchers stress the need to *understand* users**.** How is this usually done?

*HCI researchers usually understand users by observing their behaviors in focus groups and user studies.*

** Partial credit given for showing some understanding of what HCI is.*

**Question 2:** What is the cloud computing *cost associativity* idea that is so exciting? Feel free to answer this by finishing the sentence: "In the old days, you could rent 1 core for N hours for D dollars**.** Today..."

*...you can rent N cores for 1 hour for D dollars. You are able to rent any practical number of machines for a set (low) rate per core.*

** Partial credit given for mentioning the "rental" payment model where you can buy machines for an hourly rate and release them when finished.*

**Question 3:** You wish to perform a compute-intensive *mapping* task on N elements of a list, and then *reduce* all the mapper's output to a single number. It's a perfect world, and there are N cores available to help.

**a)** The _____ *programming paradigm* makes authoring this parallel code the easiest.
**b)** Given that you get ideal performance speedup, what does *Amdahl's law* tell us about the "mix" of code in your program? _____
**c)** If the order of growth of your *mapper* were LINEAR (on the size of the input list) for 1 core, then for N cores in parallel (in a perfect world) it'd be: _____
**d)** If the order of growth of your *reducer* were LINEAR (on the size of the input list) for 1 core, then for N cores in parallel (in a perfect world, where the reducer is *associative* and *commutative*, and many reducers can run at once) it'd be: _____

*a) functional*
*b) 100% parallel, 0% serial*
*c) CONSTANT*
*d) LOGARITHMIC*

** No partial credit was given for these, except in (b) where partial was given for expressing an understanding of what Amdahl's Law states.*

**Question 4:** What CS10 "big idea" was used by the BART map artists who recently redesigned the BART map to show the stops (mostly) equally spaced and in a straight line?

*Abstraction.*

** No partial credit was given.*

**Question 5:** Your final project passes the *Turing Test*, congratulations! What does that mean it can do?

*It was able to have a natural language text-only "conversation" with a human judge who couldn't tell whether it was a human or computer.*

** Partial credit was given if you described what it meant to be Turing complete, although it really doesn't have much to do with the Turing Test.*

**Question 6:** If a board game is *strongly solved* and found to be a *win*, what exactly does that mean?

*It means that the player who goes first cannot lose if they play perfectly (regardless of how smart the opponent is).*

** Partial credit for correctly stating what strongly solved OR a win means.*

**Question 7:** Recall that the *Subset Sum* problem (determining if a handful of numbers from a given set added to 0) was NP-complete (NP-hard and in NP). If a fellow CS10 student proves they've found a polynomial-time solution for it, what would that mean for a random problem in NP (say, the *Traveling Salesman* problem of a salesman who needs to find the most efficient route that goes through all cities and returns home)?

*It would mean that there was a polynomial-time solution for that one too! (since you could just reduce it to Subset Sum in polynomial time and use that answer)*

** Partial credit for expressing an understanding of the NP-complete problem space but not addressing the question at hand appropriately.*

**Question 8**: Jaron Lanier ends his "First Church of Robotics" Op-Ed piece with: *"We serve people best when we keep our religious ideas out of our work"*. Given that stance, what would he say to the IBM folks who put *Watson* on Jeopardy?

*Dispense with the histrionics and present your research in a way that educates the public about the useful of your tool in the service of humanity, not with the implication-via-theatrics "religion" that your AI will soon become our overlord.*

** A variety of different answers were accepted here as long as they related to a relevant topic mentioned in the First Church of Robotics article. Full / partial credit was given based on how well you indicated knowledge of the article and how well you linked that knowledge to Watson.*

**Question 9**: What was the remarkable achievement displayed at the "Great Robot Race" grand challenge?

*That a driverless vehicle could successfully navigate 130 miles of desert terrain (Mojave, in this case).*

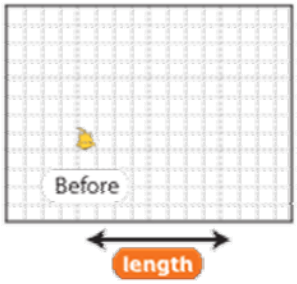
** Partial credit was given in some cases to vague but correct answers.*

Login: **cs10-____**

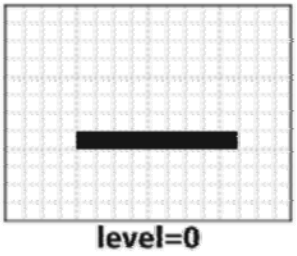
## Question 10: *Cantor Bridge over the river Kwai*

We've authored a fractal that implements the *Cantor Bridge* fractal and showed below the result of a call to

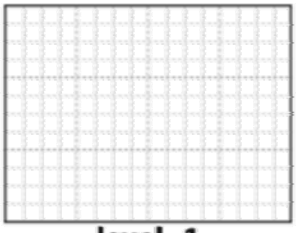

Draw the result of calls to the same block with *level* = 1 and 2**.** For each complete picture, we always start a drawing by lifting the pen (if it was down), clearing the screen, and moving the sprite to the "Before" point facing to the right. (We've made a subtle change to BYOB for this problem -- instead of a circular pen, it's a square pen. So instead of drawing lines with round endpoints, it draws lines with square endpoints.) The line segment drawn in the "*level* = 0" picture below is thickness pixels high.



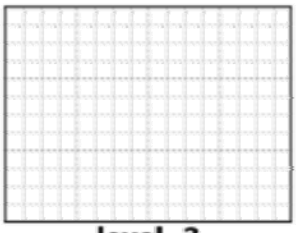




level=0

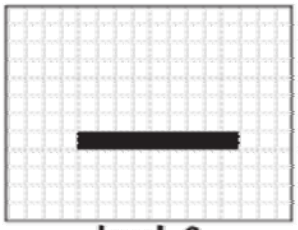


level=1

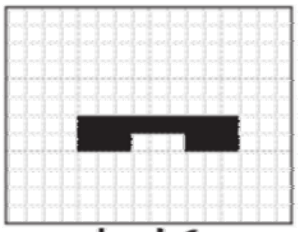


level=2

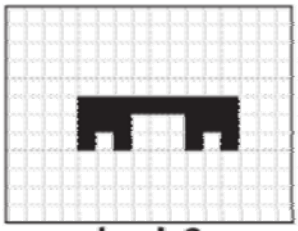
***Solution to Q10:***



level=0



level=1



level=2

** Partial credit was given for patterns that expanded properly between different levels, even if they didn't show the correct pattern.*

## Question 11: *Keep the dog away from the family tree...*

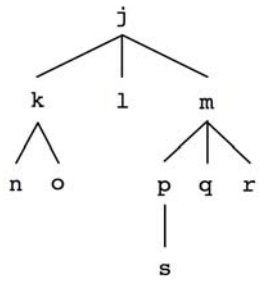
We're sure you fondly remember the ancestors person problem from the midterm. We've included it (with the answer) on the last page of this exam in case you've forgotten about it.

a. We were always bothered by the fact that we had to count *ourselves* in our ancestors. Modify the answer reprinted below so you *don't* count yourself in your ancestors. (I.e., all the ancestors person example calls should now report a number one smaller: ancestors a would report **8**, ancestors b would report **4**, etc.). You are not allowed to add any new lines, only make subtle *changes* to the code below.

```
1 ancestors(PERSON)

2   if parents-found?(PERSON)

3      report ( 1 + ancestors(father(PERSON)) + ancestors(mother(PERSON)) )

4   else

5      report ( 1 )
```

b. We would now like a list of us and all our *descendants* in no particular order (i.e., our children, their children, etc) by writing the block me and my descendants person. This problem is harder than ancestors person because we can have *many* children, but it's simpler because we only need a single helper children person which reports a list of all our children (empty if we have no kids!). Examples:

```
children j ==> (k l m)    ;; great-grandma j has 3 kids
children p ==> (s)        ;; p only has one kid, s
children l ==> ()         ;; same for n, o, q, r and s

me and my descendants j ==> (j k l m n o p q r s)
me and my descendants k ==> (k n o) ;; all these in any order
me and my descendants l ==> (l)
```




```
me-and-my-descendants(PERSON)

    if ( ____________________________________ )

        report ( ____________________________________________________ )
    else
        report ( ____________________________________________________

                 ____________________________________________________ )
```

***Solution to Q11:***

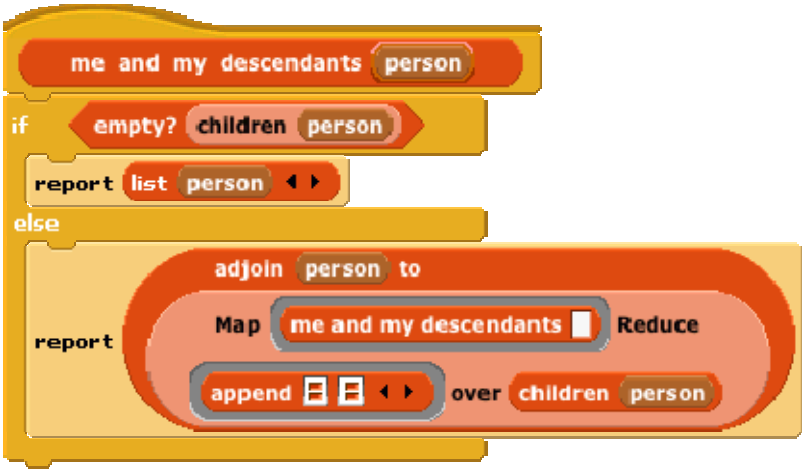
a)

```
1 ancestors(PERSON)
2   if parents-found?(PERSON)
3      report ( 2 + ancestors(father(PERSON)) + ancestors(mother(PERSON)) )
4   else
5      report ( 0 )
```

** No partial credit was given for this part.*

**b)**

```
    if ( empty?(children(PERSON)) )
      report ( list(PERSON) )
    else
      report ( adjoin(PERSON)to
                 (Map [ me-and-my-descendants ] Reduce [ append ] over (children(PERSON)))
             )
```




** Partial credit was given according to the following breakdown:*

*-1 if base case was incorrect*

*-1/2 if base case reports* ***`PERSON`*** *instead of* ***`list(PERSON)`***

*-2 if recursive case uses* ***`map`*** *but not* ***`reduce`***

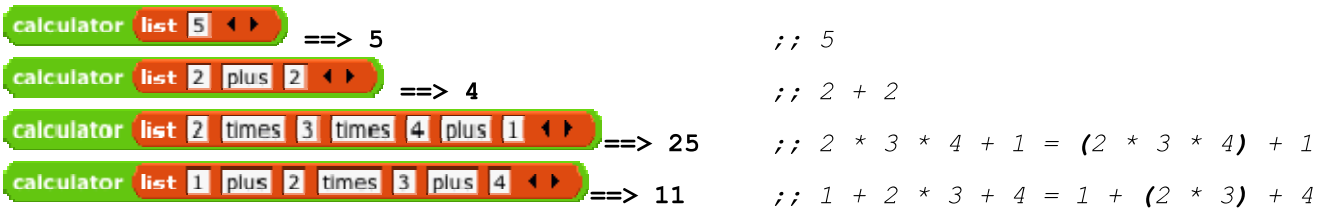
*-1 if person isn't* ***`adjoin`****'d to result of* ***`reduce`***

*-1 if* ***`map`*** *wasn't built correctly (missing / improper inputs)*

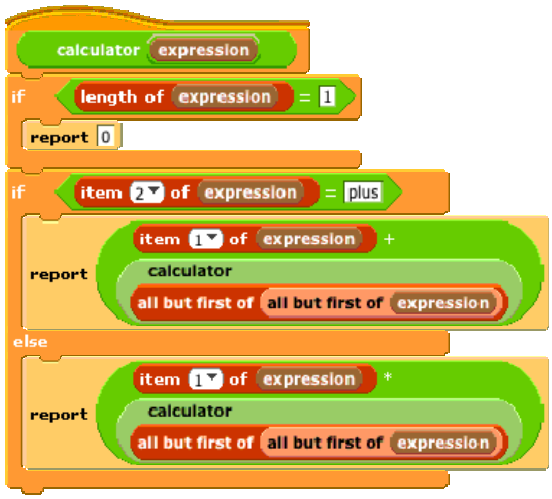
Login: **cs10-____**

## Question 12: *Driving Miss Calculate...*

You'd like to do some natural language processing, so you decide to write a reporter called **calculator** that takes an *expression* (a list) of the form **(number operation number operation … number)**, where **plus** and **times** are the only operations, and calculates the result. Examples:




Notice that **times** is always more important than **plus**. I.e., **(1 plus 2 times 3)** should behttp://61a-fa11.appspot.com/ evaluated as **1 + (2 * 3)**, and **not (1 + 2) * 3**. You may assume that **calculator** is always called on valid expressions (and is never called on an empty list). Unfortunately, your code has two bugs that you need to fix:




```
calculator(EXPRESSION)

1     if ( length-of(EXPRESSION) = 1 )
2         report ( 0 )
3     if ( item(2)of(EXPRESSION) = plus )
4         report ( item(1)of(EXPRESSION) +
5                  calculator(all-but-first-of(all-but-first-of(EXPRESSION))) )
6     else
7         report ( item(1)of(EXPRESSION) *
8                  calculator(all-but-first-of(all-but-first-of(EXPRESSION))) )
```

a. Let's first fix the most obvious bug. Replacing line # _____ with _____ would cause 

to correctly return **6** instead of **3**. The code should then work for all expressions that only use **plus**. Go ahead and make this change to the code above.

b. After you apply the fix in part (a), there's still one remaining bug. Complete the sentence:

The shortest expression that should return ___ but instead returns ___ is _____.

c. The final bug has been found in the "else" case in lines 7 and 8. Write the case correctly below. After this change, the block should work as specified on all valid expressions.

***Solution to Q12:***

a. Replacing line 2 with **`report( item(1)of(EXPRESSION) )`**

** Partial credit was given if you got one blank correct and the other incorrect.*

b. The shortest expression that should return **`3`** but instead returns **`4`** is **`(2 times 1 plus 1)`**.

** Partial credit was given for each blank that was filled in correctly.*

c. Corrected code in **bold**:

```
calculator(EXPRESSION)
   if ( length-of(EXPRESSION) = 1 )
       report ( item(1)of(EXPRESSION) )
   if ( item(2)of(EXPRESSION) = plus )
       report ( item(1)of(EXPRESSION) +
                calculator(all-but-first-of(all-but-first-of(EXPRESSION))) )
   else
       report ( calculator(adjoin(item(1)of(EXPRESSION) * item(3)of(EXPRESSION))
         to(all-but-first-of(all-but-first-of(all-but-first-of(EXPRESSION)))) )
```

** Partial credit was given according to the following breakdown:*

*-3 if the solution was missing recursion*

*-4 if there was some understanding of multiplying before recursing.*

*-5 if operations like multiplication were performed on lists.*