CSC 372 Final Exam
Monday, May 9, 2016
Solutions

Problem 1: (6 points) (mean: 5.6, median: 6)

Cite three things about programming languages you learned by watching your classmates' video projects. Each of the three should be about a different language and have a bit of depth, as described in the Piazza post that announced this problem.

When grading I tallied the video topics that were cited and came up with the counts below, with a cut-off of five. In some cases I lumped together all topics for a language into a single count.

Magic Methods in Python	20
Ruby's method_missing	17
PHP	17
JavaScript	10
Character sets in Icon	9
R	9
Swift	9
Lambdas in Java 8	8
Threads in Ruby	8
Functors in Haskell	7
Arrays in Go	6
Randomness in Haskell	6
List comprehensions in Python	6
Hotswapping in Java	5

Videos excelled in different dimensions but if I had to pick an overall winner, it would be K. Taylor's *APL - The Implications of Unicode*.

Here are some others that I felt were particularly outstanding for one reason or another. Authors are not cited because I promised the potential of anonymity.

Closures in JavaScript

Introduction to Common Lisp: S-Expressions and `list`/`quote`/`eval`

Haskell Monads in 8 Minutes

Control Structures in bash

Magic Methods in Python

Introduction to Monoids

An Exploration of Channels in Go

A Little on Functors

"Globals" in MUMPS

Python Asterisk(*) usage

Why Nobody Uses Go Arrays

Problem 2: (1 point) (mean: 0.3, median: 0)

Write a Haskell function `doflip pancakes flip` *that performs an* `fsort`*-style flip.*

```
doflip pancakes n = reverse (take n pancakes) ++ (drop n pancakes)
```

I intended this one to be a dead-simple list-manipulation problem that leveraged a concept from `fsort` but only about a third of you attempted it. The rest perhaps instinctively recoiled as soon as they saw "pancakes".

Problem 3: (3 points) (mean: 2.3, median: 3)

Write a ***recursive*** *Haskell function* `totlen list` *that returns the total length of the strings in* `list`*.*

```
totlen [] = 0
totlen (h:t) = length h + totlen t
```

Problem 4: (4 points) (mean: 2.7 median: 4)

Write a Haskell function `xout string` *that replaces every letter (a-z) in* `string` *with an "x" of the same case. Non-letters are unchanged.*

```
xout = map f
    where
        f c
            | isLower c = 'x'
            | isUpper c = 'X'
            | otherwise = c
```

Problem 5: (2 points) (mean: 1, median: 0.75)

Write a folding function `f` *such that the* `foldr` *call below behaves as shown, returning a list of the odd numbers in its last argument.*

```
> foldr f [] [5,2,9,4,4,3,1]
[5,9,3,1]

f elem acm
  | odd elem = elem:acm
  | otherwise = acm
```

Problem 6: (5 points) (mean: 4.6, median: 5)

Write a Ruby **iterator** `sbl(a, max)` ("strings by length") that first yields each one-character string in the array `a`. It then yields each two-character string in `a`. The process continues up through strings of length `max`. Strings are yielded in the order they appear in `a`. `sbl` always returns `nil`.

```
def sbl(a,max)
    for i in 1..max
        for s in a
            if s.size == i
                yield s
            end
        end
    end
    nil
end
```

Problem 7: (4 points) (mean: 2.9, median: 3)

Write a Ruby method `pages_re` that returns a regular expression that matches a string iff the string consists of one or more comma-separated page specifications. A page specification is one of these three:

- *A number, such as `"12"`.*
- *Two numbers separated by a dash, such as `"1-3"`.*
- *A number followed by a dash, such as `"2500-"`.*

```
def pages_re
    num = /\d+|\d+-\d*/
    /^#{num}(,#{num})*$/
end
```

Problem 8: (11 points) (mean: 8.8, median: 10)

Write a Ruby program `mostfreq.rb` that reads lines on standard input and writes out each line followed by an annotation that shows which non-space character appears most frequently on the line, and how many times it appears.

```
def main
    while line = gets
        line.chomp!
        puts "#{line.ljust(30)}#{count line}"
    end
end

def count s
    counts = Hash.new(0)

    s.each_char {|c| counts[c] += 1 if c != " "}
    first = counts.sort_by_value[-1]

    if first
        " ('#{first[0]}': #{first[1]})"
    else
        ""
```

```
    end
end

main
```

Problem 9: (8 points) (mean: 5.6, median: 6.5)

In this problem you are to implement a Ruby class named `Seq` that represents a sequence of values with a maximum length.

```
class Seq
    def initialize n
        @n = n
        @vals = []
    end

    def << rhs
        @vals << rhs
        if @vals.size > @n
            @vals.shift
        end
        self
    end

    def inspect
        "|" + @vals * "-" + "|"
    end
end
```

Problem 10: (6 points) (mean: 4.1, median: 5)

Write the following simple Prolog predicates. There will be a half-point deduction for each occurrence of a singleton variable or failing to take full advantage of unification.

(a) *`fl_same(?L)` expresses the relationship that the first and last elements of `L` are identical. It should able to handle all possible combinations of instantiated and uninstantiated variables. `fl_same` fails if the list is empty. Examples:*

```
fl_same(L) :- L = [H|_], last(L,H).
```

(b) *`revgen(+L,-X)` generates the elements of the list `L` <u>in reverse order</u>.*

```
revgen(L,X) :- reverse(L,RL), member(X,RL).
```

(c) Write the library predicate `member/2`. Examples:

```
member(X,[X|_]).
member(X,[_|T]) :- member(X,T).
```

Problem 11: (6 points) (mean: 3.2, median: 3)

`abl(+Atoms,+Max,-A)` first instantiates `A` to each one-character atom in `Atoms`, then each two-character atom in `Atoms`, etc. `abl` continues for atoms of lengths up through `Max`, if any are that long.

```
abl(Atoms,Max,A) :- between(1,Max,N), member(A,Atoms),
```

```
atom_chars(A,Chars), length(Chars,N).
```

Problem 12: (8 points) (mean: 4.1, median: 5)

Write a Prolog predicate `findrun(+L,+N,+X,-Pos)` that looks for `N`-long runs of `X` in `L`, instantiating `Pos` to the zero-based starting position of each run. Assume `N` is greater than zero.

```
findrun(L,N,X,Pos) :-
  repl(X,N,Mid), append(Pre,Rest,L), append(Mid,_,Rest),
length(Pre,Pos).
```

Problem 13: (7 points) (mean: 5.1, median: 5.5)

Write a predicate `expand(+List,-Expanded)` that takes a list of (1) atoms and (2) structures of the form `Atom*N` and produces an "expanded" list. Assume the `N` terms are `>= 0`.

```
expand([],[]).
expand([Atom*N|T],R)
      :- repl(Atom,N,Rep), expand(T,More), append(Rep,More,R), !.
expand([H|T],[H|R]) :- expand(T,R).
```

Problem 14: (11 points) (mean: 6.6, median: 8)

Write a Prolog predicate `reach(+Destination, +Jumps, +Fences, -Solution)` that finds a sequence of "jumps" on a Cartesian plane from `(0,0)` to the `Destination` (a `pos/2` structure), being sure that no jump lands on a fence.

```
reach(Dest, Jumps, Fences, Solution) :-
      reach0(Dest, Jumps, Fences, pos(0,0), Solution).

reach0(pos(X,Y),_,_,pos(X,Y),[]).

reach0(Dest, Jumps, Fences, pos(CX,CY), [Jump|MoreJumps]) :-
    select(Jump,Jumps,Remaining),
      Jump = jump(X,Y),
      NX is CX+X, NY is CY+Y,
      \+on_fence(pos(NX, NY), Fences),
      reach0(Dest, Remaining, Fences, pos(NX,NY), MoreJumps).

on_fence(pos(X,Y),Fences) :-
   (member(fence(x,X),Fences);member(fence(y,Y),Fences)).
```

Problem 15: (4 points) *(one point each)* (mean: 2.5, median: 2.5)

<u>The following questions and problems are related to Prolog.</u>

(1) When writing documentation for Prolog predicates, arguments are often prefixed with the symbols `+`, `-`, and `?`, such as `p(+X,?Y,-Z)`. What does each of those three symbols mean?

plus: should be instantiated
minus: will be instantiated
question mark: either instantiated or uninstantiated

(2) Write a sentence that expresses the relationship between the terms "fact", "clause", and "rule".

Facts and rules are clauses.

(3) Consider the following goal written by a novice Prolog programmer:

```
X is X + length(List)
```

*What are **two** misunderstandings evidenced by that goal?*

(1) Seems to be treating is as assignment, not unification. `X is X + anything` won't succeed unless anything is zero.

(2) Seems to be thinking of `length` not as a predicate but an arithmetic function that `is/2` knows about.

(4) What's the most important fundamental difference between a Prolog predicate like `append/3` and a function/method of the same name in Haskell, Ruby, or Java?

Prolog's `append/3` can be used to perform many operations, like taking a list apart.

Problem 16: (7 points) (mean: 6.2, median: 7)

Below is a transcript of interaction with Standard ML of New Jersey, a functional programming language. Annotate the transcript with seven significant observations about Standard ML. Excellent or additional observations may earn up to a total of three points of extra credit.

```
$ sml
Standard ML of New Jersey v110.69 [built: Mon Jun  8 23:24:21 2009]
```

A few students said, "`sml` starts a REPL". That was a little thin, but good enough!

```
- 5 + ~7;
val it = ~2 : int
```

Tilde is the unary negation operator.

Some said that tilde indicates an approximate value, and although 5 plus around 7 wouldn't be around 2, we took it, on grounds of creativity.

`int` is SML's name for an integer type.

The name `it` is bound to the last value produced.

Semicolons are required after an expression.

```
- (1, 2.0, "three");
val it = (1,2.0,"three") : int * real * string
```

There are tuples.

Tuples can be heterogenous.

`real` and `string` are names for types.

The type of tuples is shown as *`TYPE1 * TYPE2 * ... * TYPEN`*.

An ML programmer would read that type as "int cross real cross string", by the way.

```
- explode "test";
val it = [#"t",#"e",#"s",#"t"] : char list
```

`string` and `char` are distinct types. (Worth two points.)

`char` literals are `#"c"`.

List types are shown with the suffix `list`, in contrast to Haskell's `[TYPE]` notation.

Juxtaposition is function call.

```
- implode it;
val it = "test" : string
```

`implode` and `explode` are inverses.

```
- map (fn(n) => n * 3) [3, 1, 5, 9];  {- See above re map -}
val it = [9,3,15,27] : int list
```

Anonymous functions are supported.

Higher-order functions are supported.

`{- ... -}` is a comment

`[...]` is a list literal

```
- it::[];
val it = [(1,2.0,"three")] : (int * real * string) list
```

`::` is a "cons" operation.

```
- fun f1 a b = [a,b];
val f1 = fn : 'a -> 'a -> 'a list
```

`fun` is used to declare a function.

An equals sign separates the "parameters" (which could be patterns, just like Haskell) from the expression that specifies the value.

Functions are curried.

Type variables are denoted with an apostrophe. (An ML programmer would read `'a` as "alpha" and `'b` as "beta". You might have occasionally heard me inadvertently use "alpha" and "beta" when reading a type variable in Haskell.)

```
- fun f2 a b = [a = b];
val f2 = fn : ''a -> ''a -> bool list
```

The boolean type is named `bool`.

= is an equality operator.

If memory serves, only 2-3 students observed that the type variables for `f2` are preceded with two apostrophes instead of only one. I believe only student went further and correctly guessed that the double apostrophes rose from the comparison. That observation was surely

worth two points.

Background: Standard ML doesn't have the notion of type classes like `Eq`, but it does distinguish types whose values can be compared for equality, and `''a` would be recogized as an *equality type*.

```
- 3 + 4.5;
stdIn:1.1-1.6 Error: operator and operand don't agree [literal]
  operator domain: int * int
  operand:         int * real
```

"Mixed-mode" arithmetic is not allowed.

```
- (hd [1,2,3], tl [1,2,3]);
val it = (1,[2,3]) : int * int list
```

`hd` and `tl` are "head" and "tail"

I last taught Standard ML in 372 in Fall 2006. My slides are here: http://cs.arizona.edu/classes/cs372/fall06/sml.sli.pdf You'll see that a lot of ML examples translate well into Haskell examples...

Problem 17: (7 points) *(one point each unless otherwise indicated)* (mean: 3.1, median: 3.5)

Answer the following general questions.

(1) *What's something significant related to programming languages that you remember from the JarWars video we viewed and discussed during the second-to-last class?*

Tiger was the code name for Java 5, which introduced generics.

(2) *Ralph Griswold said, "If you're going to invent a language, be sure to invent a language that* you want to use*."*

(3) *What is the fundamental characteristic of a dynamically typed language?*

In general, the type of an expression can't be known without executing the code.

(4) *How does Icon avoid the "to versus through" problem that plagues string indexing in many languages and libraries?*

String positions are considered to be between characters.

(5) *What's something significant about Icon you recall from the Icon by Observation exercise during the last class? (Hint: Don't cite your answer for the previous question!)*

`*x` produces the number of elements in `x`.

(6) *With programming languages in general, what's the fundamental difference between a statement and an expression?*

Expressions produce a value; statements don't. Typically the only reason to execute a statement is to produce a side-effect.

(7) *Which of the three languages we covered this semester are you most glad we covered, and why?*

Haskell	22
Prolog	20
Ruby	9

Extra Credit Section (½ point each unless otherwise noted) (mean: x, median: y)

(1) *In as few words as possible, what is "The Cathedral and the Bazaar"? (Mentioned in `a7` solutions.)*
Book

(2) *In what month of the year 1891 were classes first held at The University of Arizona?*
October. See also
http://www.arizona.edu/topics/about-university/about-university-arizona/ua-history-and-traditions

(3) *With Prolog in mind, why is "camelcase variable" an oxymoron?*
Something like `maxValue` would be an atom!

(4) *The technology known as Leda was mentioned on Piazza. What is Leda?*
A multi-paradigm programming language created by Tim Budd.

(5) Draw an appropriate avatar for any one of the three languages we covered.
My favorite was Mr. Ferra's "Haskell-Guy", but I'm inclined to call him Lambda Man.

(6) Name a language that was once studied in depth in 372 but isn't any more.
C++, Icon, Standard ML, Lisp, Emacs Lisp are five I know of.

(7) *Who do you think whm will vote for in November's presidential election?*
There was a lot of variety here, including "nobody", "not Trump", Ralph Griswold, Mickey Mouse (an always popular write-in, election after election), and "self".

Popular guesses were these:

Bernie	9
Hillary	12
Trump	6

Early on I liked Fiorina, Cruz, Rubio, and Trump; I later came to like Kasich. But Trump's getting my vote in November!

(8) *Why did whm's solution for `pipes.pl` have the following line?* `do(p) :- do(pipes).`
So the pipes could be shown with just `p.`.

(9) *(1 point) `mostfreq.rb` says to assume that `Hash` has a `sort_by_value` method. Write Ruby code that makes that be true.*

```
class Hash
    def sort_by_value
        self.sort {|a,b| a[1] <=> b[1]}
```

```
        end
    end
```

(10) What did Knuth say about premature optimization?

It's the root of all evil.

(11) (1 point) Using only `append` and `length`, and no recursion, write a Prolog predicate `longer(A,B)`, which is true iff list `A` is longer than list `B`.

```
longer(A,B) :- length(A,N), length(X,N), append(B,[_|_],X).
    % by Patrick
```

(12) (1 point) In SNOBOL4, how do you indicate where control should go if a statement fails?

`... :F(LABEL)`

(13) whm hates writing up Piazza posts with suggested readings! They usually don't align well with his slides and he wonders if anybody actually does any of the readings. Over the course of the full semester, how many hours do you estimate you spent doing the suggested readings? (This is just data collection, no right/wrong.)

```
N = 43
mean = 3.670
median = 1.000
```

(14) Write a joke about programming languages.

A number of students expressed their true feelings about various languages with these one-word offerings:
C, Java, JavaScript, Perl, and Prolog.

I believe these are original and worth repeating:

"Prolog was invented by aliens, it's more logical than life."

"I just made a new programming language called FUN (Fantastically Underachieving Newbie!)"

"It won't right on the first run, or the second, or the third, but it might eventually."

"Prolog should be called Prolonger than Haskell."

"What did Ruby say to her boyfriend Prolog on Valentine's Day? You better HAS something good for me or I'll KELL you!"

"Why are Java programmers wealthy? Inheritance. [crickets]"

"Knock, knock.
Who's there?
Haskell.
Go away."

"My love for Prolog is TRUE."

(15) Finish this sentence: "If I only remember one thing about 372 it will be ...".

Here are a few:

"To start early."

"the FRIDAY NIGHT CLUB"

"That logic programming is virtually impossible."

"Pancakes" [several of these]

"The day you almost lost your voice during the first week of lecture." [almost?]

"Ruby is a cool language and to stay away from Prolog."

"If you are looking for a job, everyone should know [you are looking for a job]."

"append in Prolog is amazing"

"Programming problems are better when they have pancakes/food in them."

"p a n c a k e s"

"The horror of pancakes."

"Prolog programmers think Haskell is a cute toy."

"How much I love quizzes."

"1000+ slides"

"Haskell sucks"

"Ralph Griswold founded the CS department in 1971."

Statistics

All scores, in order:

```
104.00, 95.50, 95.00, 92.00, 92.00, 92.00, 91.50, 90.50, 88.50, 88.50,
87.50, 87.50, 87.50, 86.50, 86.00, 86.00, 85.00, 83.00, 81.50, 81.00, 80.50,
80.50, 79.50, 79.00, 79.00, 77.50, 77.00, 76.50, 75.00, 74.50, 74.00, 72.00,
71.50, 71.00, 70.50, 70.00, 69.50, 69.00, 68.50, 68.50, 67.50, 62.50, 62.50,
62.50, 61.00, 58.00, 58.00, 58.00, 57.50, 57.25, 57.00, 56.50, 55.50, 54.00,
50.50, 39.00, 39.00, 31.00

N = 58
mean = 73.306
median = 74.75
```