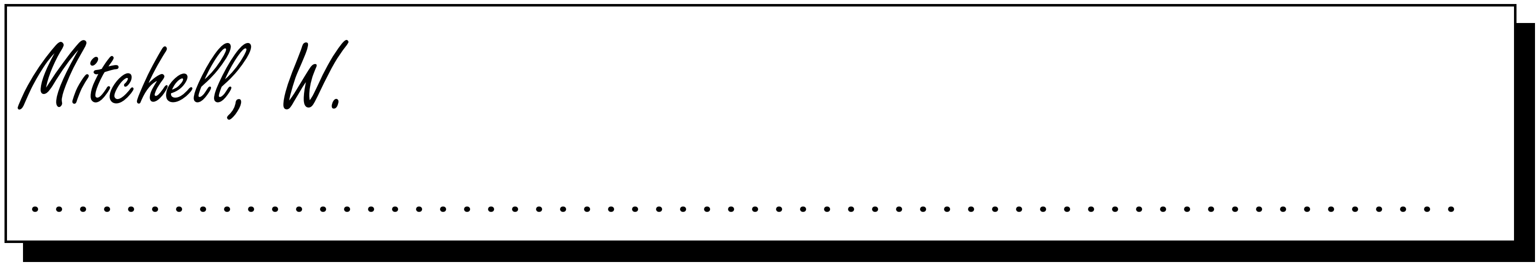
QUIZ!

Quiz Stuff

Use a full sheet of 8½x11" paper. (Half sheet? Half credit!)

Put your last name and first initial in the **far upper left hand corner** of the paper, where a staple would hit it. (It helps when sorting quizzes!)



No need to write out questions.

Numbering responses may help you avoid overlooking a question; it's ok to go ahead and pre-number your sheet.

Two minutes; five questions, plus one extra credit question.

Can everybody see this line?

Quiz 1, January 14, 2016

2 minutes; ½ point/answer; 2½ points total

1. What is the name of any one programming language created before 1975?
2. How many programming languages are there? (pick one: dozens, hundreds, thousands)
3. Who founded the UA CS department and in what year?
4. Name an area of research for which the UA CS department was recognized worldwide in the 1970s and 1980s.
5. Ideally, what percentage of your classmates will get an "A" in this course?

EC ½ point: What's the name of the most recently created language here in UA CS?

Quiz 2, January 21, 2016
3 minutes; 3 points

1. Write a Java expression that has a side effect.

2. Write a Haskell function that computes the area of a rectangle given its width and height. Append `::Int` to force it to operate on `Int`s.

3. What's the type of the function you wrote in the previous problem?

4. What does REPL stand for? Or, what's the essential functionality provided by a REPL?

5. What's a characteristic of the functional programming paradigm?

6. Imagine that `:type f` shows this: `Foo a => a -> Char`
   What does that type mean?

Solutions

1. *Write a Java expression that has a side effect.* **`x++`**

2. *Write a Haskell function that computes the area of a rectangle given its width and height. Append* **`::Int`** *to force it to operate on* **`Int`***s.* **`area w h = w * h :: Int`**

3. *What's the type of the function you wrote in the previous problem?*

   `Int -> Int -> Int`

4. *What does REPL stand for? Or, what's the essential functionality provided by a REPL?* Read-Eval-Print Loop

5. *What's a characteristic of the functional programming paradigm?*

   See slides 24-25. My quick answer: "functions are values"

6. *Imagine that* ***`:type f`*** *shows this:* ***`Foo a => a -> Char`*** *What does that type mean?*

   `f` is a function that requires a value whose type is a member of the type class **`Foo`**. **`f`** produces a **`Char`**.

Quiz 3, February 2, 2016
3 minutes; ½ + ½ + 2 points

1. Add parentheses to the following expression to show the order of operations: **`a b + x y z`**

2. The **`length`** function produces the length of a list. What's the type of **`length`**?

3. Write a function **`nzs`** that returns the number of zeroes in a list. (2 points!)

```
> nzs [5,0,0,5]
2
```

EC ½ point:
Write a function **`f`** whose type is inferred to be **`a -> a -> a`**. Be sure that **`a`** doesn't have a class constraint, like **`Eq a`**.

Solutions

1. *Add parentheses to the following expression to show the order of operations:* ***a b + x y z***

```
(a b) + ((x y) z)
```

2. *The* ***length*** *function produces the length of a list. What's the type of* ***length****?* **[a] -> Int**

3. *Write a function* ***nzs*** *that returns the number of zeroes in a list.*

   Two solutions:

```
nzs [] = 0
nzs (0:t) = 1 + nzs t
nzs (_:t) = nzs t

nzs [] = 0
nzs (h:t)
    | h == 0 = 1 + nzs t
    | otherwise = nzs t
```

EC ½ point: Write ***f*** *whose type is inferred to be* ***a -> a -> a****.*

```
f x y = head [x, y]
```

Quiz 4, February 9, 2016

3 minutes; ½ + ½ + 2 points

1. Give a simple definition for "higher order function".

2. What's the type of **`map`**? Here's a reminder of how **`map`** works:

```
> map (add 2) [1..5]
[3,4,5,6,7]
```

3. Write a function **`atb f x y`** that calls the function **`f`** with the larger of **`x`** and **`y`**. (2 points!)

```
> atb negate 7 2
-7

> atb length "aa" "zzz"
3
```

EC ½ point: In Haskell, what's a "section"? (Ok to just show an example.)

Solutions

1. *Give a simple definition for "higher order function".*

   A function that has one or more arguments that are functions.

2. *What's the type of* **`map`**?

   ```
   (a -> b) -> [a] -> [b]
   ```

3. *Write a function* **`atb f x y`** *that calls the function* **`f`** *with the larger of* **`x`** *and* **`y`**. *(2 points!)*

   Two solutions:

   ```
   atb f x y = f (if x > y then x else y)

   atb f x y
       | x > y = f x
       | otherwise = f y
   ```

EC ½ point: In Haskell, what's a "section"? (Ok to just show an example.)

Short answer: (+3) is a section.

Long answer: A syntactic mechanism that allows creation of a partial application of a binary operator by supplying either operand.

Quiz 5, February 18, 2016
3 minutes; 2 + ½ + ½ points

1. Consider folding a list of strings into the total length of the strings:

   ```
   > foldl f 0 ["just", "a", "test"]
   9

   > foldl f 0 ["abc"]
   3
   ```

   For this problem you are to write a folding function **`f`** that would work as shown with the **`foldl`** calls above

2. What's a difference between **`foldl`** and **`foldr`**?

3. What's a difference between **`foldr`** and **`foldr1`**?

Solutions

1.
```
> let f acm elem = acm + length elem

> foldl f 0 ["just", "a", "test"]
9

> foldl f 0 ["abc"]
3
```

2. What's a difference between ***`foldl`*** *and* ***`foldr`****?*

`foldl` folds the list from left to right but **`foldr`** folds from right to left.

3. What's a difference between ***`foldr`*** *and* ***`foldr1`****?*

`foldr` needs a "zero" value, for the case of an empty list.

Quiz 6, February 25, 2016
3 minutes; 1 + (½ * 6) points

1. What's a fundamental characteristic of a statically typed language? (one point)

2. Name a language that uses static typing.

3. Name a language that uses dynamic typing.

4. Assuming **`s = "abcdef"`**, what's the value of **`s[1,3]`**?

5. Assuming **`s = "abcd"`**, what's the value of **`s[1..-1]`**?

6. What's a key difference between Ruby arrays and Haskell lists?

7. What program provides a REPL for Ruby?

EC: Who invented Ruby?

Solutions

1. *What's a fundamental characteristic of a statically typed language? (one point)*

   The type of every expression can be determined without running the code.

2. *Name a language that uses static typing.* Java, Haskell, C

3. *Name a language that uses dynamic typing.* Ruby, Python, Icon

4. *Assuming* ***`s = "abcdef"`****, what's the value of* ***`s[1,3]`****?* **`"bcd"`**

5. *Assuming* ***`s = "abcd"`****, what's the value of* ***`s[1..-1]`****?* **`"bcd"`**

6. *What's a key difference between Ruby arrays and Haskell lists?*

   Ruby arrays are heterogeneous—they can hold a mix of types.

7. *What program provides a REPL for Ruby?* **`irb`**

EC: Who invented Ruby? Yukihiro Matsumoto ("Matz" ok!)

Quiz 7, March 8, 2016

3 minutes; ½ + ½ + ½ + ½ + 1 points

1. How can we quickly tell whether an identifier is a global variable?
2. In Ruby's world, what is an iterator?
3. What keyword does an iterator use to invoke a block?
4. What element of Haskell is a Ruby block most like?
5. Assuming **`x`** is an array, print the elements in **`x`**, one per line, using the iterator **`each`** with an appropriate block.

E.C. ½ point: Write a trivial iterator.

Solutions

1. *How can we quickly tell whether an identifier is a global variable?* Starts with a dollar sign, like **`$x`**.

2. *In Ruby's world, what is an iterator?*
   An iterator is a method that can invoke a block.

3. *What keyword does an iterator use to invoke a block?* **`yield`**

4. *What element of Haskell is a Ruby block most like?*
   An anonymous function, I'd say.

5. *Assuming* ***`x`*** *is an array, print the elements in* ***`x`****, one per line, using the iterator* ***`each`*** *with an appropriate block.*

```
x.each {|e| puts e}
```

E.C. ½ point: Write a trivial iterator.

```
def itr
   yield 7
end
```

Quiz 8, March 22, 2016

3 minutes; ½ point each

1. What Ruby operator looks to see if a string contains a match for a regular expression?

2. When that matching operator is successful, what value does it produce?

3. After a successful match, what does the predefined global **`` $` ``** hold? (dollar-backquote)

4. What's the longest string that can be matched by the RE **`/a..z/`**?

5. Languages vary in their level of support for regular expressions. What level of support does Ruby provide—library support or syntactic support?

6. Support your answer for the previous question with a brief explanation.

Solutions

1. *What Ruby operator looks to see if a string contains a match for a regular expression?* **`=~`** is the "match" operator.

2. *When that matching operator is successful, what value does it produce?* The starting position of the first match

3. *After a successful match, what does the predefined global* ***`$``*** *hold? (dollar-backquote)*

   The portion of the string preceding the match

4. *What's the longest string that can be matched by the RE* ***`/a..z/`****?*

   Four characters

5. *Languages vary in their level of support for regular expressions. What level of support does Ruby provide—library support or syntactic support?* Syntactic support

6. *Support your answer for the previous question with a brief explanation.*

   The expression ***`/text/`*** designates a regular expression.

Quiz 9, March 22, 2016
4 minutes; 2 + 1½ + ½ points

1. In English, describe strings that are matched by this Ruby regular expression: **`/^[xyz]+[abc]?\d{2,3}/`**

2. Write a definition for a Ruby class named **`X`**. Instances of **`X`** are created by specifying a string, like **`X.new("abc")`**. **`X`** has one method, named **`f`**, that returns the length of the string that the instance was created with.

3. The line "**`attr_reader :x`**" in a class definition specifies that there should be a getter for the instance variable **`x`**. What's especially interesting about **`attr_reader`**?

EC ½ point: Briefly, what's the difference between a language being extensible vs. being mutable?

Solutions

1. *Describe strings matched by* ***`/^[xyz]+[abc]?\d{2,3}/`***

   Starts with one or more occurrences of **`x`**, **`y`**, or **`z`**; followed by an optional **`a`**, **`b`**, or **`c`**; followed by two or three digits.

2. *Write a definition for a Ruby class named* ***`X`***. ...

```
class X
    def initialize s
        @f = s.size
    end
    attr_reader :f
end
```

3. *The line "**`attr_reader :x`**" in a class definition specifies that there should be a getter for the instance variable **`x`**. What's especially interesting about **`attr_reader`**?*

   `attr_reader` is a method that generates a getter method.

EC ½ point: If a language is mutable, the meanings of operations can be changed, where extensibility only allows for providing meaning for previously undefined operations.

Quiz 10, March 29, 2016

45 seconds; ½ point each

With Prolog in mind...

1. Write an example of an atom.

2. Write an example of a fact.

3. Write an example of a query.

EC: What's the command to run SWI Prolog on lectura?

Solutions

1. *Write an example of an atom.* **`food`**

2. *Write an example of a fact.* **`food(apple).`**

3. *Write an example of a query.* **`food(apple).`**

Shorter:

```
a
a(b).
a(b).
```

EC: What's the command to run SWI Prolog on lectura?

```
swipl
```

Quiz 11, April 12, 2016

4 minutes; ½ + 1 + ½ + 1 + ½ + ½ points

1. Write an example of a structure with two terms.
2. What are two distinct computations that can be done with the predicate **`food/1`** that we've been using?
3. What does the notation **`f/3`** mean?
4. Draw and label the ports of the four-port model.
5. What is the output of the following query?

   ```
   ?- A=1, B=2, write(A), A=B, write(B).
   ```

6. Given these facts, **`a(1). a(1,2). a(2).`**
   what is output by the following query?

   ```
   ?- a(X), writeln(X), fail.
   ```

Solutions

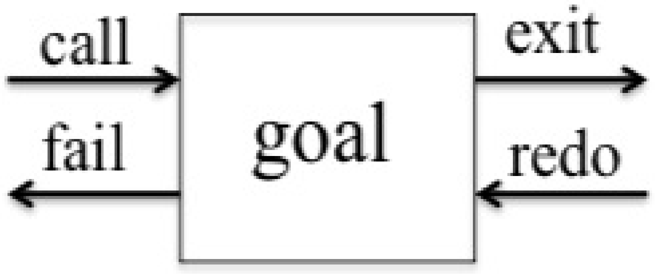
1. *Write an example of a structure with two terms.* **`x(1,2)`**

2. *What are two distinct computations that can be done with the predicate **`food/1`** that we've been using?*

   (1) Ask if something is food. (2) Enumerate all foods.

3. *What does the notation **`f/3`** mean?* `f` is a three-term predicate.

4. *Draw and and label the ports of the four-port model.*



5. *What is the output of the following query?*

```
?- A=1, B=2, write(A), A=B,
   write(B).
1
false.
```

6. *Given these facts,* **`a(1). a(1,2). a(2).`** *what is output by the following query?*

```
?- a(X), writeln(X), fail.
```

```
1
2
false.
```

Quiz 12, April 19, 2016
3 minutes; 1 + 2 points

1. Write a predicate **`same(?A, ?B, ?C)`** that expresses the relationship that **`A`**, **`B`**, and **`C`** are equal.

```
?- same(1,1,1).
true.

?- same(a,X,a).
X = a.
```

2. Write a predicate **`p(+L)`** that prints the elements of **`L`** that are integers, one per line. (Use **`integer(?X)`** to test.) <u>Be sure it always succeeds!</u>

```
?- p([10,b,c,2,4]).
10
2
4
true.
```

Solutions

1. Write a predicate **`same(?A, ?B, ?C)`** that expresses the relationship that **`A`**, **`B`**, and **`C`** are equal.

```
same(X,X,X).
```

2. Write a predicate **`p(+L)`** that prints the elements of **`L`** that are integers, one per line.

```
p(L) :- member(E,L), integer(E),
        writeln(E), fail.
p(_).
```

Quiz 13, April 26, 2016

4 minutes; 4 points

1. What would Prolog show for **A** and **B** for the folllowing query?
   ```
   ?- [_,A|B] = [1,2,3,4].
   ```

2. Without using **append**, write **head(List, Elem)**.

3. Without using **append**, write **last(List, LastElem)**.

4. What are two predicates that can be used, respectively, to add or remove facts?

EC ½ point: Write **g(L,E)** that generates each element of **L** twice:

```
?- g([a,b],E).
E = a ;
E = a ;
E = b ;
E = b ;
false.
```

Solutions

1. *What would Prolog show for* **A** *and* **B** *for the folllowing query?*
```
?- [_,A|B] = [1,2,3,4].
A = 2, B = [3, 4].
```

2. *Without using* **append***, write* **head(List,Elem)***.*
```
    head([H|_],H).
```

3. *Without using* **append***, write* **last(List, LastElem)***.*
```
    last([X],X).
    last([_|T],X) :- last(T,X).
```

4. *What are two predicates that can be used, respectively, to add or remove facts?* **assert** and **retract**

EC: Write **g(L,E)** *that generates each element of* **L** *twice.*
```
    g([H|_],H).
    g([H|_],H).
    g([_|T],E) :- g(T,E).
```

Quiz 14, April 28, 2016
4 minutes; 4 points

1. Briefly describe the general approach used to solve the pit-crossing puzzle in the slides.

2. Write a predicate **`inc`** that uses **`assert`** and **`retract`** to increment a counter maintained as a **`count/1`** fact. It reports the new value.

```
?- count(N).
N = 0.

?- inc.
Count is 1
true.

?- inc.
Count is 2
true.

?- count(N).
N = 2.
```

Solutions

1. *Briefly describe the general approach used to solve the pit-crossing puzzle in the slides.*

   Pick a plank from the supply. See if it can be placed without ending over a pit. If so, solve it from there using the remaining planks. If not, pick a different plank and try again.

2. *Write a predicate* **`inc`** *that uses* **`assert`** *and* **`retract`** *to increment a counter maintained as a* **`count/1`** *fact. It reports the new value.*

```
inc :-
   count(N0),
   retract(count(_)),
   N is N0+1,
   assert(count(N)),
   format('Count is ~w~n', N).
```