CMSC 330 Exam 1 - Study Guide

Topics:

- Programming Language Concepts
- Ruby Introduction (basics, data structures, etc.)
- Ruby Regular Expressions
- Ruby Codeblocks
- OCaml Introduction (expressions, functions, types, basics, etc.)
- OCaml Lists and Pattern Matching
- OCaml Lets, Tuples, and Records
- OCaml Higher-order Functions
- OCaml Datatypes
- General idea of finite state machine (no NFA/DFA)

Notes:

- Dynamic typing: check at runtime (Python, Ruby, JavaScript,...)
- Static typing: check at compile time (Java, C, Ocaml,...)

=> Ruby is dynamic, and Ocaml is static

- Manifest or Explicit: have to tell the types of new variables (Java, C,...)
- Latent or Implicit: No need (Ruby, Ocaml, Python)

=> Ruby and Ocaml are latent

1. Programming Language Concepts

Spring 2018

1.1 **[7 pts]** Circle the correct answer:

a. *True* / ***False***: [1,2,3] is a list/array of three ints in both OCaml and Ruby

b. ***True*** / *False*: Static type checking occurs at compile time

c. ***True*** / *False*: In dynamically typed languages, a type error will go unnoticed if the line containing the error is never executed

d. The OCaml compiler does which of the following if you omit a case in a pattern match: *Nothing* / ***Emits a warning*** / *Emits an error*

e. *True* / ***False***: Ruby variables are declared explicitly

f. ***True*** / *False*: All values in Ruby are objects

g. *True* / ***False***: Ruby code blocks are *first class*, e.g., they can be stored in arrays

Explain:
- a is True because [1,2,3] is array of 3 ints in Ruby but not in Ocaml, it's array of a 3-tuple
- b was mentioned
- c is True since dynamic type is checking at runtime, meaning error won't be raised if error line isn't executed
- d => no ideas
- e => ...
- f is True because code-blocks are not values
- g is False since code-blocks aren't objects, so can't be stored in arrays

Fall 2018

Programming Language Concepts

1. [1 pts] (T / F) In every programming language, code must be compiled before it is run.

   Solution. False. ❑

2. [1 pts] (T / F) Static typing occurs during program execution and dynamic typing occurs before the program is run.

   Solution. False. ❑

In the following questions, circle **all** answers that apply.

3. [2 pts] In Ruby, which of the following are objects?

   (a) `true` (b) `Hash.new` (c) `2` (d) `[1]`

   Solution. (a), (b), (c), (d) ❑

4. [2 pts] In OCaml, which of the following are true about functions?

   (a) They can take other functions as arguments.

   (b) They have to be given a name to be used.

   (c) They will throw an error if not given enough arguments.

   (d) They can return another function as an output.

   Solution. (a), (d) ❑

5. [2 pts] Which of the following is stored in a closure?

   (a) the execution stack (b) the function's output (c) the environment (d) the function's code

   Solution. (c), (d) ❑

6. [2 pts] Which of the following fit the functional programming paradigm?

   (a) loops

   (b) recursion

   (c) higher-order functions

   (d) mutable variables

   Solution. (b), (c) ❑

1 => False since Ruby isn't compiled before it is run
2 => ... 3 => ...
4 => (b) is false because we have anonymous function
(c) is false since it can return another function
5 => no idea (we don't need to know about closure)
6 => no loops and mutable variables in Ocaml

Spring 2019

1. [10 pts] Programming Language Concepts

Circle your answer

A. Tuples in OCaml are similar to structs in C in that they are both fixed-sized collections of heterogeneous data. (**T** / F)

B. Ruby has type inference for its variables. (T / **F**)

C. In dynamically typed languages, type errors may go unnoticed if they are inside rarely used conditional branches. (**T** / F)

D. A let...in expression in Ocaml is used to define a named local expression. (**T** / F)

E. Because of dynamic type checking, Ruby allows programs with type errors to run. (**T** / F)

F. Ruby arrays can hold different objects and dynamically resizable. (**T** / F)

G. Both Procs in Ruby and functions in OCaml have "first class" status; e.g., they can be passed to and returned from methods/functions. (**T** / F)

H. If two objects are structurally equal, they must be physically equal too. (T / **F**)

I. A closure consists of function code and bindings for its free variables. (**T** / F)

J. Compiled languages typically run slower than interpreted languages because of the extra overhead of converting source code to machine code at runtime. (T / **F**)

Notice: H & J

Fall 2019

Circle your answer. Each question is 1 point.

T	F	qsort in C is a higher order function because it takes a function pointer as an argument
T	**F**	In OCaml, `[1]::[2]` is equivalent to `[1;2]`.
T	**F**	OCaml type inference occurs at runtime.
T	F	In Ruby, `x = "apple"; y = x;` is an example of a reference copy.
T	**F**	OCaml tuples are homogeneous.
T	**F**	Structural equality implies physical equality.
T	**F**	For a statically-typed language, you have to specify the type of variables when declaring them.
T	F	Functions in OCaml are first class.
T	F	Ruby supports implicit variable declarations.
T	**F**	Ruby code blocks are first class.

Spring 2020

T	F	In statically typed languages, a type error will go unnoticed if the line containing the error is never executed.
T	F	Immutability is a key concept of functional languages.
T	F	Ruby code blocks are first-class; e.g., they can be passed into and returned from methods, and can be assigned to variables directly.
T	F	Closures are used to implement dynamic scoping.
T	F	OCaml uses a static type checking system.
T	F	If a programming language has type inference, then variable types are ignored until runtime.
T	F	The `map` function is an example of a higher order function.
T	F	In all languages with static type checking, the variable type must be explicitly declared.
T	F	The cons operator combines two lists together in OCaml.
T	F	Higher order functions refer to functions that take in other functions as arguments or return a function.

Fall 2020

Q2.1 Ruby Objects

4 Points

In Ruby, which of the following is an object? Check all that apply.

- **A string**
- **An integer**
- **`nil`**
- **A class**

Q2.2 Ruby Arrays vs Hashes

3 Points

Briefly describe one difference between Arrays and Hashes in Ruby. Which would be more suitable for storing a sorted collection of values?

Arrays are indexed by consecutive integers from 0, hashes are indexed by anything. An array would be more suitable because a hash orders elements by the order in which they were inserted, but an array allows you to insert elements anywhere in the list.

Spring 2021: Skipped (Nothing helpful)
Fall 2021: Skipped (Nothing helpful)
Spring 2022:

Q2.1. A single programming language can be compiled or interpreted, or both. T / F

Q2.2. In Ruby, the following is a type error that is caught before runtime: `"a" + 1` T / F

Q2.3. You cannot have a tuple of functions in OCaml, that is a tuple of the following signature:
`('a -> 'a) * ('b -> 'b)` T / F

Q2.4. `[7, 8, 9]` is valid in both Ruby and OCaml. T / F

Q2.5. In the given Ruby code, which of the following are true? Select all that apply.

```
a = [1, 2, 3]
b = [1, 2, 3]
(a.equal? b && a == b)
```

- `equal?` Returns `false` because a and b are not "structurally" equal
- `==` returns `true` because a and b are "structurally" equal
- The expression returns `false`
- The expression returns `true`

Summer 2022 (Some NFA/DFA questions):

Q2.4
2 Points

A Finite State Machine (Finite Automata) can be used to check if an arbitrary string is a palindrome

- ○ True
- ◉ False

Save Answer ***Unsaved Changes**

Q2.8
3 Points

Name one advantage a DFA has over an NFA

You will always know which state you are in when traversing a DFA

Why would we want to treat functions as data, like we do in OCaml?

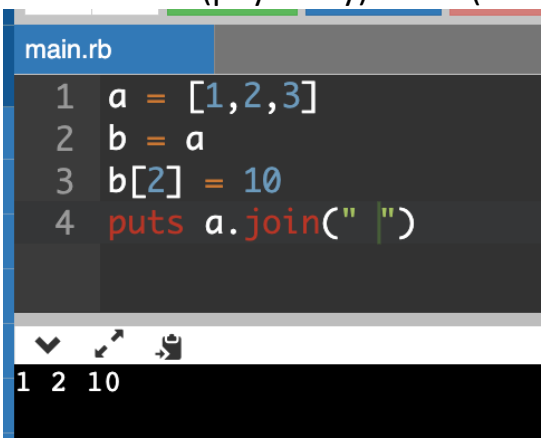
Treating functions as data allow for variety of positives:
lambda expressions
currying
code resuability

Fall 2022: Skipped (Nothing helpful)

Summary of our unknown things:

- Structurally equal and physically equal:
    + Structural qualities concern the current contents of the arguments only. Structural comparisons are therefore more straight forward: compare the contents of the arguments
    + Physical qualities concern the physical property. Two things are physically equal if and only if changing one changes the other.

in Ruby: equals? (physically) and == (structurally) when comparing strings
In Ocaml: == (physically) and = (structurally)



=> a and b are physically equal

So, structurally equality doesn't imply physically equality, but physically equality can imply structurally equality

- [1,2,3] = [(1,2,3)] in Ocaml
- Again, static/dynamic => when type checking occurs
    manifest/latent => how we determine type of variables
    Those 2 aren't related to each other. Like when saying "For a statically-typed language, you have to specify the type of variables when declaring them" isn't true because static typing does type checking at compiled time. Ocaml is a counter-example => static but latent (no need to declare variable types)
- Tuples are fixed-sized/heterogeneous (more than 1 type)
- Compiled languages do not run slower than interpreted languages
- Key concept of functional languages:
    + immutability
    + recursion
    + higher-order functions
    + no side-effects
    + first-class functions??? => A programming language is said to have First-class functions when functions in that language are treated like any other variable.
    => Ocaml has first-class functions, Ruby doesn't but has procs
    (Closures concepts, do not need to know :D)

==>> Carefully read the questions, words may be changed but the concepts shouldn't be new

2. Ruby Regular Expressions

Regex quick reference

[abc]	A single character of: a, b, or c	.	Any single character	(...)	Capture everything enclosed
[^abc]	Any single character except: a, b, or c	\s	Any whitespace character	(a|b)	a or b
[a-z]	Any single character in the range a-z	\S	Any non-whitespace character	a?	Zero or one of a
[a-zA-Z]	Any single character in the range a-z or A-Z	\d	Any digit	a*	Zero or more of a
^	Start of line	\D	Any non-digit	a+	One or more of a
$	End of line	\w	Any word character (letter, number, underscore)	a{3}	Exactly 3 of a
\A	Start of string	\W	Any non-word character	a{3,}	3 or more of a
\z	End of string	\b	Any word boundary	a{3,6}	Between 3 and 6 of a

options: i case insensitive m make dot match newlines x ignore whitespace in regex o perform #{...} substitutions only once

3. Ruby Coding/Code-blocks

- Remember how to use yield, proc call, array.each, hash.each, file.each... (Read my Ruby notes)
- How to define a function, class => instance variables, class variables, remember @ and @@ when using those variables
- Distinguish indexing array/hash

main.rb

```ruby
a = [1,2,3]
a[10] = 10
b = {1=>"2"}
puts b["232"].class
puts a.length
puts a["232"]
```

input

```
NilClass
11
main.rb:6:in `[]': no implicit conversion of String into Integer (TypeError)
        from main.rb:6:in `<main>'
```

4. Ocaml Typing/Basics

expressions => types

types => expressions

Finding errors:

- unbound variables: variables have not been given any value
- incorrect type for the if condition: must be bool

- mismatched return types
- mismatched types in function calls
- etc.

Remember that if bool then 'a else 'a

```
match 'a with
    |'a -> 'b
    |'a -> 'b
    ....
```

5. Ocaml Coding:

- Know how to use pattern matching for list/tuple/record/variant (check my Ocaml notes)
- Know how to use recursion to create a new list/variant (linked list/tree)

6. Ocaml higher-order functions:

- Use map when it's simple, output list having same length as input list
- Fold: consider using fold_left or fold_right properly
- Remember to use tuple for accumulator when needed, then use match (...) -> ... to extract the required fields. For example, using the 1 more variable in acc as a current index.
- When dealing with list, consider using ::(cons) and @ correctly to get the right order for output

7. General idea of finite state machine:

Read 1.1 & 1.2 & 1.5 on Cliff's notes (https://bakalian.cs.umd.edu/assets/notes/FA.pdf)

1.3 & 1.4 & 1.6 will be on quiz 3