Final Exam

Student Name
Search students by name or email...

Q1
0 Points

Please carefully read the instructions below:

Ground Rules

This exam is open-note, which means that you may refer to your own notes and class resources during the exam. However, doing so will cause you to have less time to complete the exam. You can also use irb and utopt (or other programs). You may not work in collaboration with anyone else, regardless of whether they are a student in this class or not. If you need to ask a question about the exam, post a private question on Piazza.

Sections

<table>
<thead>
<tr>
<th>Section</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL Concepts</td>
<td>[20 pts]</td>
</tr>
<tr>
<td>Ruby Code</td>
<td>[8 pts]</td>
</tr>
<tr>
<td>Ruby Coding</td>
<td>[8 pts]</td>
</tr>
<tr>
<td>OCaml Code</td>
<td>[8 pts]</td>
</tr>
<tr>
<td>Section</td>
<td>Points</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------</td>
</tr>
<tr>
<td>OCaml Coding</td>
<td>8 pts</td>
</tr>
<tr>
<td>OCaml Coding</td>
<td>8 pts</td>
</tr>
<tr>
<td>FSMs</td>
<td>8 pts</td>
</tr>
<tr>
<td>Grammars</td>
<td>8 pts</td>
</tr>
<tr>
<td>OpSem</td>
<td>7 pts</td>
</tr>
<tr>
<td>Lambda Calc</td>
<td>8 pts</td>
</tr>
<tr>
<td>Rust Coding</td>
<td>3 pts</td>
</tr>
<tr>
<td>Rust COde</td>
<td>8 pts</td>
</tr>
<tr>
<td>Syntax vs Semantics</td>
<td>6 pts</td>
</tr>
</tbody>
</table>

**General Advice**

You can complete answers in any order, and we recommend you look through all of the questions before first so you can gauge how long you should spend on each question. Refer to the counter in the top left corner to ensure you have completed all questions.

**Submission**

You have 120 minutes to complete this exam (see the timer in the upper right corner for remaining time). Once you begin, you can submit as many times as you want until your time is up. You can even leave this page and come back, and as long as the time hasn’t expired, you’ll be able to update your submission. This means that if you accidentally submit, refresh, or lose internet temporarily, you’ll still be able to work on the test until the time is up. If you come back, click "Resubmit" in the bottom-right corner to resume.
Honor Pledge

Please copy the honor pledge below:

I pledge on my honor that I have not given or received any unauthorized assistance on this examination.

Enter your answer here

Signature

By entering your name below, you agree that you have read and fully understand all instructions above.

Enter your answer here

Q2 PL Concepts

20 Points

Q2.1

2 Points

Lexers don’t care if the input is grammatically incorrect

☐ True

☐ False

Save Answer *Unsaved Changes

Q2.2

2 Points

Not all programming languages are turing complete
Q2.3
2 Points
Not all NFAs can be represented by a regular language
- True
- False

Q2.4
2 Points
Any set of strings that a Regular Expression describes can also be described by a CFG
- True
- False

Q2.5
2 Points
Rust's ownership rules help prevent memory issues found in other languages like C
- True
- False
Q2.6
2 Points
Operational Semantics focus on describing a program based on mathematical objects
- True
- False

Save Answer  *Unsaved Changes

Q2.7
2 Points
Keywords like `mut`, `int`, `void*`, describe segments of memory, rather than how the compiler/interpreter should treat the data.
- True
- False

Save Answer  *Unsaved Changes

Q2.8
3 Points
Choose Ruby, Ocaml or Rust, and describe why we would want to use that language over the other 2.

answers will vary

Save Answer  *Unsaved Changes

Q2.9
3 Points
Why do so many languages exist when we could just use one?
answers will vary

Q3 Ruby Code
8 Points

Q3.1
2 Points

Consider the following Ruby Code:

```ruby
result = []
myhash = {"one"=>[11,20,8], "two"=>[17,12], "three"=>[17,8,4]}
myhash.each { |k,v|
  sum = 0
  v.each { |x| sum+=x}
  result.append(sum/v.length())
}
puts result
```

What is the output?

13
14
9

Q3.2
2 Points

Consider the following:
def function(a,b):
    arr = [a,b]
    if a > 10
        yield arr
    else
        yield [10,10]
    end
end

function(1,2) {__Blank 1__}
function(11,1) {__Also Blank 1__}

What could we replace `Blank 1` with so that the following is printed?

```
20
20
22
2
```

```
|x,y|
puts x*2
puts y*2
```

Q3.3
4 Points

Consider the following code:

```python
a = "Android 2B, Deployed at Location: Desert along with 9S"
b = "Android A2, Deployed at Location: City Ruins along with 2P"
re = Regexp.new(/^Android \d[A-Z], Deployed at Location: ([a-zA-Z])\s+\d+/)
hash = {}
for i in [a,b]
    if i =~ re
        puts $1
        hash[$1] = [$2,$3]
end
end
```
Why does this not print "Not deployed in same area" (ie. Why is the guard on line 11 true)?

$2 and $3 don't exist in the regex so they return nil, making the hashes the same

Change a single line from lines 3-10 such that "2B\nA2\nNot deployed in same area" is printed

change line 3 to
re = Regexp.new("^Android (\d[A-Z][A-Z]\d), Deployed at Location: ([[a-zA-Z]\s]+) along with (\d[A-Z]\$)"")

Q4 Ruby Coding
8 Points

Let's write an interpreter. Instead of a AST though, you will be given arrays to represent data.

Here is the grammar

\[
R \rightarrow \ R + \ R \\
\ | \ R - \ R \\
\ | \ R \ast \ R \\
\ | \ R/\ R \\
\ | \ (R) \\
\ | \ n
\]

where \( n \) is any integer.

Here is an array that represents a sentence:
Each array will be a size of 3

- The first element of an array will be either "$add","sub","mult","div"
- The second element will be either an array or string of an integer. It will represent the left expression of the operator
- The third element will be either an array or string of an integer. It will represent the left expression of the operator

You can assume that you will be passed in a grammatically correct sentence

- Division will be integer division
- You may want to use .class

examples:

eval ["add","3","4"] => 7
eval ["sub", ["mult","1","5"],"7"] => -2
eval ["div", ["add","4","6"],["sub","5","0"] ] # 2

def eval expr

if expr[0] == "add" then (eval expr[1]) + (eval expr[2])
elsif expr[0] == "sub" then (eval expr[1]) - (eval expr[2])
elsif expr[0] == "mult" then (eval expr[1]) * (eval expr[2])
elsif expr[0] == "div" then (eval expr[1]) / (eval expr[2])
else expr.to_i
end

e

Q5: Ocaml Code

https://www.gradescope.com/courses/399941/assignments/2124402/submissions/new
8 Points

**Q5.1**
2 Points

Give an expression of the type `a -> b -> c`. All pattern matching must be exhaustive.

```
let rec f a b = f a b
```

*Unsaved Changes

**Q5.2**
2 Points

```
fold (\_blank 1\_) 0 [(1,2,3);(4,5,6)]
```

Fill in the blank such that the code segment will return the sum of second item each tuple in the list. For example, the above code should return 7.

**Blank 1**

```
fun a (_,b,\_) -> a + b
```

*Unsaved Changes

**Q5.3**
4 Points

```
1 let rec f x y =
2   match x with
3     [] -> 0
4     [(a,b)] -> let (c,d) = y in c +. d
5     (a,b)::t -> a + b
```
Why will this code not compile?

```
type error, expecting int, but line 4 gives a float
```

Rewrite a single line from 3-5 so that it does.

```
change line 4 to
[(a,b)] -> let (c,d) = y in c + d
```

**Q6 OCaml Coding**

8 Points

Given two lists of integers ranging from zero to infinity, write a function that sums together items at the same index and then returns the largest of these sums.

**NOTE:** You may only use map and fold provided above and declare any helper function. Do not use `rec` keyword for the function itself, but the helper function may be recursive.

- You may assume that two lists are going to be of equal length.
- When both lists are empty, just return 0

Examples:

```
largest_sum [1;2;3;4;5] [10;11;12;13;14] = 19
largest_sum [] [] = 0
largest_sum [5;7] [4;3] = 10
```

```
let largest_sum lst1 lst2 =

  let rec helper l1 l2 =
  match (l1,l2) with
  ([],[]) -> [0]
  |(h1::tl1),(h2::tl2) -> (h1+h2)::(helper tl1 tl2) in
  fold (fun a b -> if b > a then b else a) 0 (helper lst1 lst2);
```
Q7 FSM
8 Points

Use this NFA for the following questions:

Q7.1
2 Points

What is the regex of the machine?

(alb?c)((albc?)(ab?c))*
Q7.2
6 Points

I attempted to use NFA to DFA but I am missing some things.

Using the naming conventions we used in the class, give the name of the state(s) I am missing separated by semicolons if more than 1

[0,1,3]

Using the syntax from the project (eg: ([0],"a",[2])), give the transitions missing.

[[0,2,1],"c",[0,1,3]]
[[0,1,3],"a",[0,1]]
[[0,1,3],"b",[0,2,1]]
[[0,1,3],"c",[c]]

Which states should be marked as final states?
Q8 Grammars
8 Points

Q8.1
4 Points

Provide two derivations with all steps shown that prove the following CFG is ambiguous.

\[
\begin{align*}
S & \rightarrow AB \\
A & \rightarrow AaA \mid a \\
B & \rightarrow BB \mid b
\end{align*}
\]

\[
\begin{align*}
S \rightarrow AB \rightarrow AaAB \rightarrow aaAB \rightarrow aaAaAB \rightarrow aaaaAB \rightarrow aaaaaB \\
S \rightarrow AB \rightarrow AaAB \rightarrow AaAaAB \rightarrow aaAaAB \rightarrow aaaaAB \rightarrow aaaaaB
\end{align*}
\]

Rewrite the following CFG so that it still represents the same language but is no longer ambiguous.

\[
\begin{align*}
S \rightarrow AB \\
A & \rightarrow aaAaaa \\
B & \rightarrow bBlb
\end{align*}
\]

Q8.2
4 Points

Define a CFG that describes the language.
$a^x b^y c^z$ where $z = x + 2y$, $x \geq 0$ and $y > 0$.

**Note:** To represent $e$ in the CFG, you can either copy and paste the symbol $e$, type the word *epsilon* or just type the letter $e$.

\[
S \rightarrow aScIT \\
T \rightarrow bTcIbCc
\]

Q9 Opsem
7 Points

Consider the following OpSem Rules:

\[
\begin{align*}
A; \text{true} & \Rightarrow \text{true} & A; \text{false} & \Rightarrow \text{false} \\
A; \text{e}_1 & \Rightarrow \text{true} & A; \text{e}_1 & \Rightarrow \text{false} \\
A; (\text{not e}_1) & \Rightarrow \text{false} & A; (\text{not e}_1) & \Rightarrow \text{true} \\
A; \text{e}_1 & \Rightarrow \text{true} & A; \text{e}_2 & \Rightarrow v_1 \\
A; (\text{if e}_1 \text{ then e}_2 \text{ else e}_3) & \Rightarrow v_1 \\
A; \text{e}_1 & \Rightarrow \text{false} & A; \text{e}_3 & \Rightarrow v_1 \\
A; (\text{if e}_1 \text{ then e}_2 \text{ else e}_3) & \Rightarrow v_1 \\
A; \text{e}_1 & \Rightarrow v_1 & A; \text{e}_2 & \Rightarrow v_2 & v_3 \text{ is } v_1 \text{ } \&\& \text{ } v_2 \\
A; (\text{e}_1 \text{ } \&\& \text{ } \text{e}_2) & \Rightarrow v_3
\end{align*}
\]

Fill in the following derivation:
\[ A; \ #4 \Rightarrow \ #4 \quad A; \ #5 \Rightarrow \ #5 \quad \#6 \text{ is } \#3 \]
\[ A; \ #1 \Rightarrow \ #1 \quad A; \ #3 \Rightarrow \ #6 \quad A; \ #2 \Rightarrow \ #7 \]

if \( \text{false} \) then \( \text{true} \) else not \((\text{true} \&\& \text{false})\) \(\Rightarrow\) \#7

Blank 1

false

Blank 2

not(\text{true} \&\& \text{false})

Blank 3

\text{true} \&\& \text{false}

Blank 4

true

Blank 5

false

Blank 6

false

Blank 7

true

*Unsaved Changes

Q10 Lambda Calc
8 Points
Q10.1 Encodings
4 Points

Consider the following encodings,

true = (λx. λy. x)
false = (λx. λy. y)
not = (λx. x false true)
if = (λx. λy. λz. x y z)

Prove that if (not true) false true = true

Note: You must make all parenthesis explicit before reducing the expression.

(not true) false true
(λx. x false true true) false true
(λx.( x false true true)) false true
(true false true) false true
(λx. λy. x false true) false true
( λy.false true) false true
(false) false true
(λx. λy. y) false true
λy y true
true

Q10.2 Variables
2 Points

Consider the following Lambda expression
λx. y x (λy. y y x)(λx. x y x) x
If we label each variable from left to right like so
λx₀. y₀ x₁ (λy₁. y₂ y₃ x₂)(λx₃. x₄ y₅ x₆) x₆.

Which variables are the free variables?

y₀, y₄
Q10.3 CBV and CBN
2 Points

Consider the following Lambda expression
\((\lambda x. x y) \lambda x. x (\lambda x. (\lambda y. y) x)\)

Evaluate following expression in both call by value and call by name. Make sure to make all parenthesis explicit and show all alpha conversion. If it cannot be further reduced, write "Cannot be reduced"

Call By Value:

\[
(\lambda x. (x y)) (\lambda x. (x (\lambda x. (\lambda y. y) x)))
(\lambda x. (x y)) (\lambda a. (\lambda b. ((\lambda c. c) b)))
(\lambda x. (x y)) (\lambda a. (\lambda b. b))
(\lambda a. (\lambda b. b)) y
(y (\lambda b. b))
\]

Call By Name:

\[
(\lambda x. (x y)) (\lambda x. (x (\lambda x. (\lambda y. y) x)))
(\lambda x. (x y)) (\lambda a. (\lambda b. ((\lambda c. c) b)))
(\lambda a. (\lambda b. (\lambda c. c) b)) y
(y (\lambda b. (\lambda c. c) b))
(y (\lambda b. b))
(y (\lambda b. b))
\]

Q11 Rust Coding
3 Points
Given two descendingly sorted integer vectors, write merge that returns a vector that merged the two inputs in descending order.
Examples:

```rust
fn merge(v1: Vec<i32>, v2: Vec<i32>) {
    let mut ret = Vec::new();
    for i in v1.iter(){
        ret.push(*i)
    }
    for i in v2.iter(){
        ret.push(*i)
    }
    ret.sort();
    return ret
}
```

Q12 Rust Code
8 Points

Q12.1
2 Points

Consider the following Rust Code:

```rust
fn main () {
    let mut a = String::from("Hello World");
    let b = a;
    let c = &b;
```
function1(c);
println("{}",c);
function2(b);
println("{}",b);
}

Does the following program compile? If so, write out the output. Otherwise, point out the line that causes the error and explain the error.

function2 takes in b, and then println("{}",b); tries to use b after move.
It does not compile

Q12.2
4 Points

Consider the following Rust Code

fn main() {
    let mut a = 42;
    let b = &mut a;
    let &mut d = b;
    let e = a;
    let c = *d;
    let f = &e;
}

Who owns the int 42 when the function ends?

e

How many borrows were there?

3
Q12.3
2 Points

Consider the following Rust Code

```rust
struct Rectangle{
    width:i32,
    height:i32,
}

impl Rectangle{
    fn perimeter(&self) ->i32{
        return self.width*2 + self.height*2
    }
}
```

Write an implementation block with one associated function called `perimeter` which gives the perimeter of a rectangle.

Q13 Semantics and Syntax
6 Points

Consider the following C code

```c
#include <stdio.h>
int mystery(int x){
    return x+1;
}
int main()
{
    int arr[5] = {1,2,3,4,5};
    int ret = 0;
    for (int i = 0; i < 5; i++){
        ret += mystery(arr[i]);
    }
    printf("ret: %d",ret);
}
```
Consider what this code segment does. Without simplifying the program (getting rid of any unnecessary), convert the code.

```
int x = 3;
printf("Hello");
```

would be converted to the following in java

```
int x = 3;
System.out.println("Hello");
```

**Q13.1**

2 Points

Convert this code segment to Ruby:

```
def mystery(x)
  x + 1
end

def main
  a = [1,2,3,4,5]
  ret = 0
  for i in a
    ret += mystery(i)
  end
  puts("ret: "+ ret.to_s)
end
```

**Q13.2**

2 Points

Convert this code segment to OCaml:
let mystery x = x + 1 in
let main = let a = [1;2;3;4;5] in
let rec helper l = match l with
  [] -> 0
  h::t -> (mystery h) + (helper t) in
let ret = helper a in
let _ = print_string "ret: " in
print_int ret in main

Q13.3
2 Points

Convert this code segment into Rust:

```rust
fn mystery(x:i32)->i32{
    x+1
}
fn main()
{
    let arr = [1,2,3,4,5];
    let mut ret = 0;
    for i in arr.iter(){
        ret += mystery(*i)
    }
    println!("ret: ",ret);
}
```

*Unsaved Changes*