Final Exam from Fall 2020 (Practice)

STUDENT NAME

Search students by name or email...

Q1 Introduction

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. 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

- PL Concepts [8pts]
- Lambda Calculus [8pts]
- OCaml [15pts]
- Ruby [12pts]
- Rust [8pts]
- Language Representation [15pts]
- Parsing [12pts]
- Operational Semantics [12pts]
- Security [10pts]

General Advice

You can complete answers in any order, and we urge you to look through all of the questions at the beginning so you can accurately 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.

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

Save Answer

Q2 PL Concepts

8 Points

Q2.1 Role of a Lexer

2 Points

A lexer converts a program's source code into an abstract syntax tree.

- ○ True
- ○ False

Save Answer

Q2.2 Functional Paradigms

2 Points

The functional programming paradigm prefers immutable values.

- ○ True
- ○ False

Save Answer

Q2.3 Static vs Dynamic Typing

4 Points

What is one benefit of using a statically-typed language?

Enter your answer here

What is one benefit of using a dynamically-typed language?

Enter your answer here

Save Answer

Q3 Lambda Calculus
8 Points

Q3.1 Alpha-Equivalence
2 Points

Are the following λ-expressions alpha equivalent?

$(\lambda x.x\ y)$ vs $(\lambda w.w\ v)$

- Yes
- No

Save Answer

Q3.2 Call-By-Value vs Call-By-Name
6 Points

Consider the following λ-expression: $(\lambda a.a\ a)\ ((\lambda b.c)\ (\lambda d.c\ e))$. Show all steps for full points.

Reduce this expression using call-by-value:

Enter your answer here

Reduce the expression using call-by-name:

Enter your answer here

Save Answer

Q4 OCaml
15 Points

Q4.1 Write Function of Type

3 Points

Write a function of the type `'a list -> 'b list -> ('a * 'b * 'b)`. The function must not contain any non-exhaustive pattern matching, but you are allowed to raise exceptions.

Enter your answer here

Save Answer

Q4.2 Recursive `count`

3 Points

Write a function called `count` that takes in a value and a list and returns the number of times the value occurs in the list (as determined by `=`). You **may not** use functions from the `List` module, and you **may not** use map or fold. The function can be recursive. You may not define any helper functions.

For example:

```
count 1 [1; 1; 3; 2; 1] = 3
count 'a' ['b'; 'a'; 'a'; 'c'] = 2
count 5 [] = 0
```

The function header should be `let rec count x lst =` (include this in your answer)

Enter your answer here

Save Answer

Q4.3 Non-recursive `count`

3 Points

Now re-write the function `count` from the previous question so that it uses `List.fold_left`. This function must not be recursive. You **may not** use any functions from the `List` module. You may not define any helper functions.

You may use `map` or `fold`, given below:

```
let rec map f l =
    match l with
    | [] -> []
    | h :: t -> (f h) :: (map f t)

let rec fold f acc l =
    match l with
    | [] -> acc
    | h :: t -> fold f (f acc h) t
```

The function header should be `let count x lst =` (include this in your answer)

Enter your answer here

Save Answer

Q4.4 Total Length of list list

6 Points

Write a function called `total_size` which takes in a `'a list list` and returns the total number of elements in all of the sublists. You may not define helper functions, but you can use `map` or `fold` which are given above. It is up to you whether to make the function recursive.

For example:

```
total_size [] = 0
total_size [[]; []] = 0
total_size [[1]; [2]; [3; 5]] = 4
total_size [[1] [2; 10; 2]; []; [3; 5]; []] = 6
```

The function header should be `let rec total_size lst =` (include this in your answer, `rec` is optional)

Enter your answer here

Save Answer

Q5 Ruby

12 Points

Tom Nook has left you stranded on a deserted island after you failed to make a single mortgage payment in 10 years. Your only hope to survive the night is to build a house. You are given a file containing a list of items and for each item and for each item, the resources needed to craft the item.

For example, the file might look like this:

```
Chair: 3 wood, 1 ore, 1 stone
Table: 2 stone, 2 wood
Roof: 4 wood, 1 ore, 1 stone
Door: 3 wood
Foundation: 20 wood, 40 stone, 10 ore
```

Format specifications:

- The only three materials are "wood", "ore" and "stone", but *can be in any order*
- Each material can appear at most once on each line
- Each item will have at least one material
- The item (e.g. Chair, Table, etc) can be any word of length at least 1 which begins with an uppercase letter and is followed by zero or more lowercase letters
- All numbers are positive integers without leading zeros

The format will be exactly as above in the example; that is there will be no extra whitespace before, after, or in between parts. **Assume all input you are given is valid, following this format exactly**. We aren't trying to trick you.

Q5.1 `parse_file`

6 Points

Write a function called `parse_file` which takes a filename and returns a hash where the keys are the items, and each value is a list of sublists, where each sublist is of the form `[number, material]`. For example, calling `parse_file` with the above example file should return

```
{
    "Chair" => [ [3, "wood"], [1, "ore"], [1, "stone"] ],
    "Table" => [ [2, "stone"], [2, "wood"] ],
    "Roof" => [ [4, "wood"], [1, "ore"], [1, "stone"] ],
    "Door" => [ [3, "wood"] ],
    "Foundation" => [ [20, "wood"], [40, "stone"], [10, "ore"] ],
}
```

The order of the outer lists don't matter.

You should use the starter code below and replace the part marked `TODO`:

```
def parse_file(filename)
    recipe_hash = {}
    File.readlines(filename).each do |line|
        # TODO
    end
    return recipe_hash
end
```

Enter your answer here

Save Answer

Q5.2 `total_cost`

6 Points

Write a function called `total_cost` which takes a list of materials and a hash like that created in the previous part, and returns the total price of constructing all the items in the list. The costs of wood is 3, the cost of ore is 9, and the cost of stone is 2.

For example:

```
total_cost(["Chair",  "Roof", "Roof", "Foundation"], hash_from_part_1) = 270
total_cost([], hash_from_part_1) = 0
```

You can assume the hash you are given is correct even if you did not complete part 1, and that all items in the list are present in the recipe hash.

The function signature should be `def total_cost(items, recipe_hash)`

Enter your answer here

Save Answer

Q6 Rust
8 Points

Q6.1 Ownership
4 Points

Consider the following snippet of Rust code:

```
let a = String::from("ferris");
let b = String::from("rustacean");
let c = &a;
let d = b;
let e = c;
```

Which variable currently owns the string "ferris"?

- `a`
- `b`
- `c`
- `d`
- `e`

Which variable currently owns the string "rustacean"?

- `a`
- `b`
- `c`
- `d`
- `e`

Save Answer

Q6.2 Ownership and Borrowing

4 Points

Consider the following snippet of Rust code:

```
fn main() {
    let a = String::from("cmsc330");
    let b = &a;
    {
        let mut c = a;
        c.push_str(" rocks!");
        println!("{}", c);
    }
    /* HERE */
}
```

At the line marked "HERE", which of the following is true?

- ○ `a` owns the string "cmsc330 rocks!"
- ○ `b` owns the string "cmsc330 rocks!"
- ○ The string has been dropped
- ○ This code does not compile

Save Answer

Q7 Language Representation

15 Points

Q7.1 Construct a CFG

3 Points

Construct a CFG which accepts strings of the form $a^x bc^x d$, where $x \geq 1$.

Enter your answer here

Save Answer

Q7.2 Ambiguous CFG

4 Points

Prove that the following grammar is ambiguous by showing two distinct derivations of the same string.

$$S \rightarrow \mathbf{a}S \mid S\mathbf{b} \mid T$$
$$T \rightarrow \mathbf{c}T \mid \mathbf{c}V$$
$$V \rightarrow V\mathbf{b} \mid \varepsilon$$

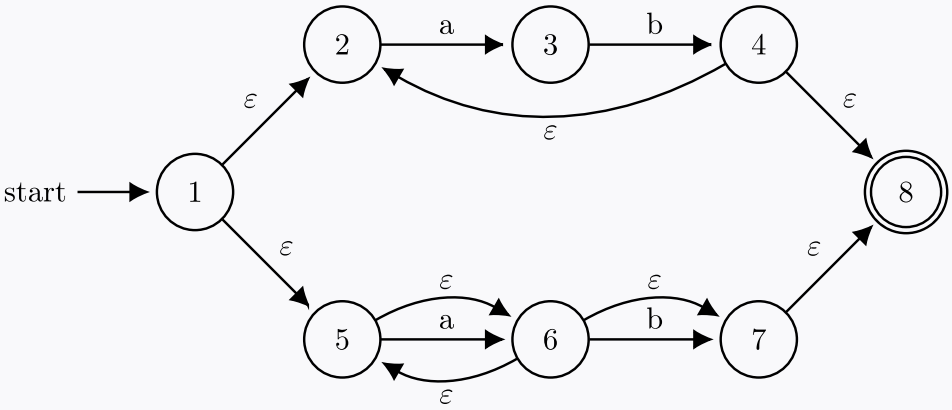
Enter your answer here

Save Answer

Q7.3 NFA to Regex

4 Points

Write a regular expression which is equivalent to the following NFA:



Enter your answer here

Save Answer

Q7.4 CFG to Regex

4 Points

Write a regular expression that accepts the same set of strings as the following CFG:

$S \rightarrow A\mathbf{c}$
$A \rightarrow \mathbf{c}A \mid \mathbf{c}B$
$B \rightarrow \mathbf{ab}B \mid \mathbf{a}$

Enter your answer here

Save Answer

Q8 Parsing

12 Points

Q8.1 Recursive Descent Parsing

3 Points

Consider the following CFG:

$S \rightarrow \mathbf{n}ST \mid TC\mathbf{x}$
$T \rightarrow \mathbf{d}T \mid \varepsilon$
$C \rightarrow C\mathbf{f} \mid \varepsilon$

Can the above grammar be parsed using a recursive descent parser? Briefly justify.

Enter your answer here

Save Answer

Q8.2 Write a Parser

9 Points

Consider the following CFG:

$S \rightarrow \mathbf{let}\ T = E\ \mathbf{in}\ S \mid E$
$T \rightarrow \mathbf{a} \mid \mathbf{b}$
$E \rightarrow T \mid \mathbf{n}$

Using only the following helper functions, implement a parser for the above grammar. For our purposes, the keywords **let** and **in** can each be treated as a single token. So the list of valid tokens are: "**let**", "**in**", "=", "**a**", "**b**", "**n**". You must use the functional approach (like project 4), not the imperative approach.

The following functions are given:

```
exception ParseError of string

let lookahead toks : string =
    match toks with
    | [] -> ""
    | h::_ -> h

let match_tok toks (a : string) =
    match toks with
    | h::t when a = h -> toks
    | _ -> raise (ParseError "bad match")
```

Each parser function should only return the updated list of tokens.

Write `let rec parse_S toks` below:

Enter your answer here

Write `let rec parse_T toks` below:

Enter your answer here

Write `let rec parse_E toks` below:

Enter your answer here

Save Answer

Q9 Operational Semantics

12 Points

Q9.1 Proof

8 Points

Using the following rules:

$$\frac{}{A; n \to n} \qquad \frac{A(x) = v}{A; x \to v} \qquad \frac{A; e_1 \to v_1 \qquad A, x : v_1; e_2 \to v_2}{A;\ \textbf{let}\ x = e_1\ \textbf{in}\ e_2 \to v_2}$$

$$\frac{A; e_1 \to n_1 \qquad A; e_2 \to n_2 \qquad n_1 < n_2}{A; e_1 < e_2 \to \textbf{true}} \qquad \frac{A; e_1 \to n_1 \qquad A; e_2 \to n_2 \qquad n_1 \geq n_2}{A; e_1 < e_2 \to \textbf{false}}$$

$$\frac{A; e_1 \to n_1 \qquad A; e_2 \to n_2 \qquad v = n_1 - n_2}{A; e_1 - e_2 \to v} \qquad \frac{A; e_1 \to n_1 \qquad A; e_2 \to n_2 \qquad v = n_1 + n_2}{A; e_1 + e_2 \to v}$$

Fill in the blanks in the proof below:

$$\dfrac{\dfrac{}{A; 8 \to 8} \qquad \dfrac{\dfrac{\dfrac{\boxed{1}}{A, x : 8; x \to 8} \qquad \dfrac{}{\boxed{2}} \qquad 4 = 8 - 4}{\boxed{3}} \qquad \dfrac{}{A, x : 8; \boxed{4}} \qquad \boxed{5}}{A, x : 8; \boxed{6} \to \textbf{true}}}{A;\ \textbf{let}\ x = 8\ \textbf{in}\ 3 < x - 4 \to \textbf{true}}$$

Blank 1:

Enter your answer here

Blank 2:

Enter your answer here

Blank 3:

Enter your answer here

Blank 4:

Enter your answer here

Blank 5:

Enter your answer here

Blank 6:

Enter your answer here

Save Answer

Q9.2 myst Operator

4 Points

Given the following operational semantics rules for **myst**, which logical operator does **myst** represent?

$$\frac{A; e_1 \to \textbf{true} \qquad A; e_2 \to \textbf{true}}{A; e_1 \ \textbf{myst} \ e_2 \to \textbf{false}} \qquad \frac{A; e_1 \to \textbf{true} \qquad A; e_2 \to \textbf{false}}{A; e_1 \ \textbf{myst} \ e_2 \to \textbf{true}}$$

$$\frac{A; e_1 \to \textbf{false} \qquad A; e_2 \to \textbf{true}}{A; e_1 \ \textbf{myst} \ e_2 \to \textbf{true}} \qquad \frac{A; e_1 \to \textbf{false} \qquad A; e_2 \to \textbf{false}}{A; e_1 \ \textbf{myst} \ e_2 \to \textbf{false}}$$

You can write the name of the operator, or briefly describe what it does.

Enter your answer here

Save Answer

Q10 Security

10 Points

Q10.1 Type Safe Languages

1 Point

Type safe languages prevent command injection.

- True
- False

Save Answer

Save Answer

Q10.2 Cookies

1 Point

Servers use cookies in order to keep track of users who have already logged in.

- ○ True
- ○ False

Save Answer

Q10.3 XSS

1 Point

Bob posts the following HTML/JavaScript on a social media website which does not escape input:

```
<script>alert("Hello!");</script>
```

Later, Alice visits the website and Bob's code is executed, causing an alert dialog saying "Hello!" to pop up on her screen. This is an example of:

- ○ Stored XSS
- ○ Reflected XSS

Save Answer

Q10.4 Escaping

1 Point

Which of the following vulnerabilities can be prevented through escaping?

- ○ Command injection
- ○ XSS
- ○ SQL Injection
- ○ All of the above

Save Answer

Q10.5 Analyzing Code

6 Points

A multiplayer game allows players to enter their own usernames for each match, which can be seen by other players. That username is then checked against a list of banned usernames in an SQL database. If the username is not banned, then it is linked to the `user_id` in the `user_table`. **The values in `user_table` are later shown to other players using a javascript-**

coded leaderboard. Below (in Ruby) is the function they use to ask the user for their preferred username.

```ruby
def get_username(user_table, user_id)
    while true do
        puts "Enter your username: "
        username = gets
        results = @db.execute "SELECT * FROM BannedUsers WHERE Name = '#{username}';"
        if results.nil? then
            user_table[user_id] = username
            break
        else
            puts "Banned username detected!!! Try again"
        end
    end
end
```

Name a vulnerability that exists in this code [2 points]

Enter your answer here

Give an example of an input that would exploit this vulnerability [4 points]

Enter your answer here

Save Answer

Save All Answers

Submit & View Submission