Exam 2 from Spring 2021 (Practice)

STUDENT NAME

Search students by name or email...

Q1 OCaml

12 Points

The following are questions about OCaml programming.

Q1.1 References

4 Points

OCaml references introduce which of the following problems (check all that apply)?

- [ ] immutability
- [ ] evaluation order matters
- [ ] cyclic datastructures could induce infinite loops
- [ ] cannot always substitute `f(x)+f(x)` with `2 * f(x)`

Save Answer

Q1.2 References

3 Points

Fill in the blank such that the the following expression will evaluate `z` to be `18`:

```
let x = ref 3 in
let y = !x in
let () = __________ in
let z =  !x * y in
z
```

Enter your answer here

Save Answer

Q1.3 Closures

2 Points

Consider the following code:

```
let x = 2 in
let f = fun x -> fun y -> x-y in
let g = f 3 in
...
```

What is the code portion of the closure stored in `g`?

- `fun x -> fun y -> x-y`
- `x-y`
- `fun y -> x-y`
- `fun y -> 2-y`

Save Answer

Q1.4 Closures

3 Points

In Q1.3, what is the value of `x` in `g`'s closure's environment, which will be used when `g` is called?

- 2
- 3
- 5
- There is no x in the environment

Save Answer

Q2 NFA/DFA

27 Points

Q2.1

2 Points

NFA string `accept` (e.g., in Project 3) is usually faster than DFA `accept`.

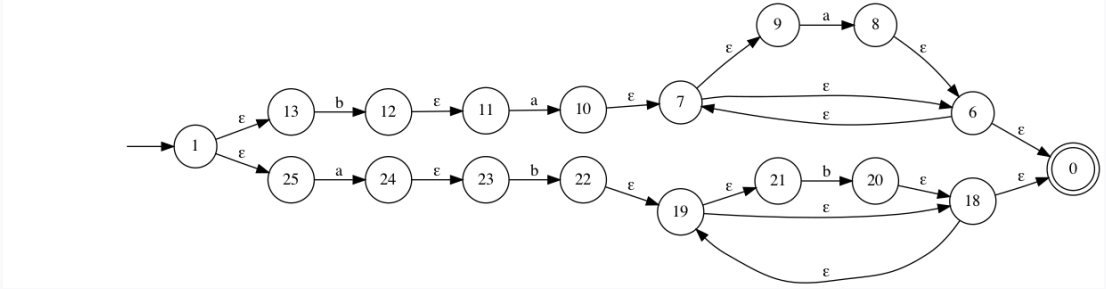
- True
- False

Save Answer

Q2.2 Regexes and NFAs

6 Points

Given a regular expression that is equivalent to the following NFA.



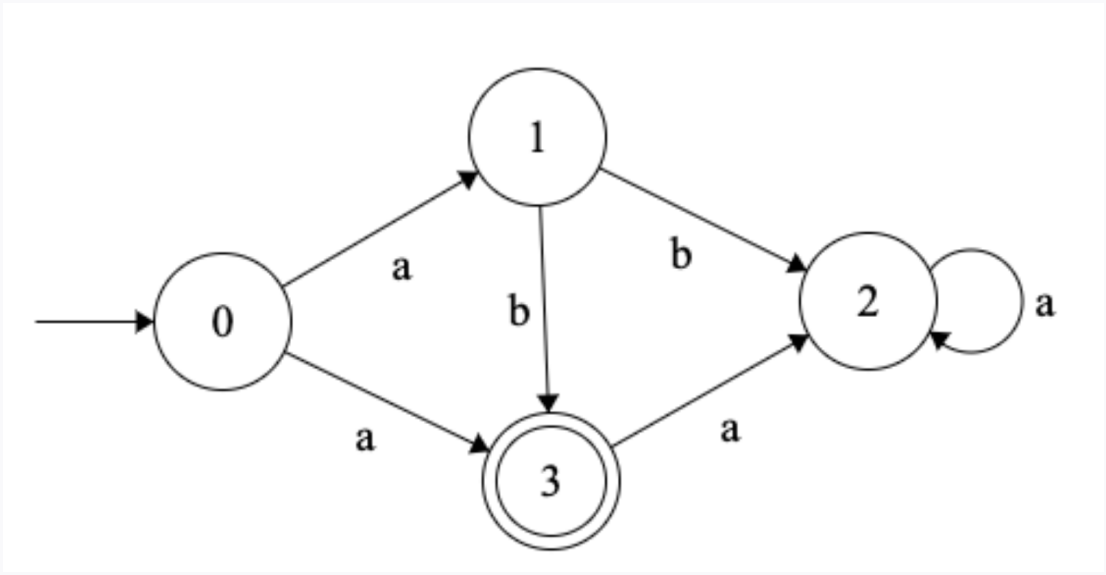
Enter your answer here

Save Answer

Q2.3 NFA acceptance

4 Points

Consider the following NFA:



Which of the following strings will be accepted by this NFA? Check all that apply. (You might want to look at questions 2.4-2.7 before answering this question.)

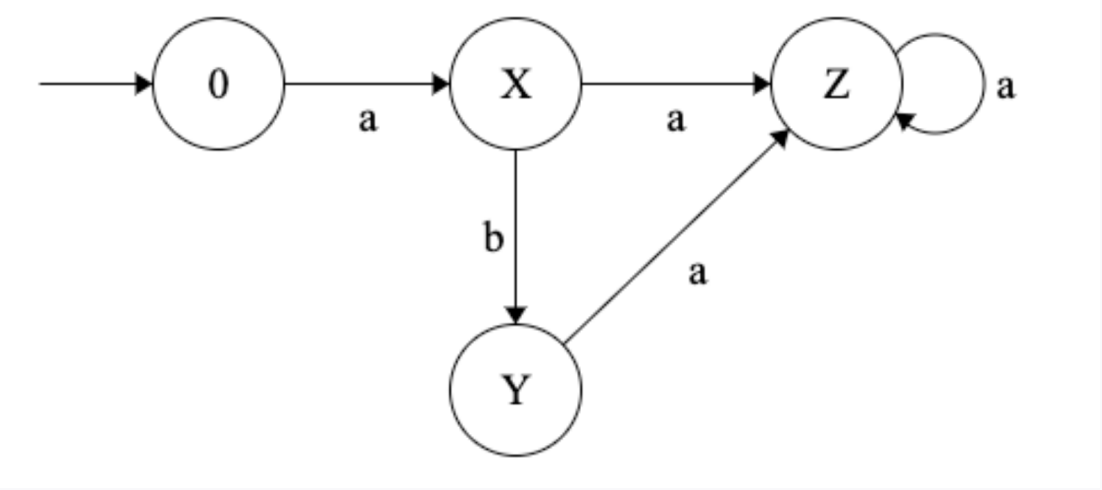
- [ ] a
- [ ] aaaaaaa
- [ ] abab
- [ ] abaaa

Save Answer

Q2.4 NFA->DFA conversion (part 1)

3 Points

The NFA from Q2.3 can convert to the following equivalent DFA, using the subset construction:



In this DFA, which NFA states make up the set X (check all that apply)?

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4

Save Answer

Q2.5 NFA->DFA conversion (part 2)

3 Points

In the converted DFA in 2.4 above, which NFA states are in the set Y (check all that apply)?

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4

Save Answer

Q2.6 NFA->DFA conversion (part 3)

3 Points

In the converted DFA in 2.4 above, which NFA states are in the set Z (check all that apply)?

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4

Save Answer

Q2.7 NFA->DFA conversion (part 4)

3 Points

In the converted DFA in 2.4 above, which DFA states are final states?

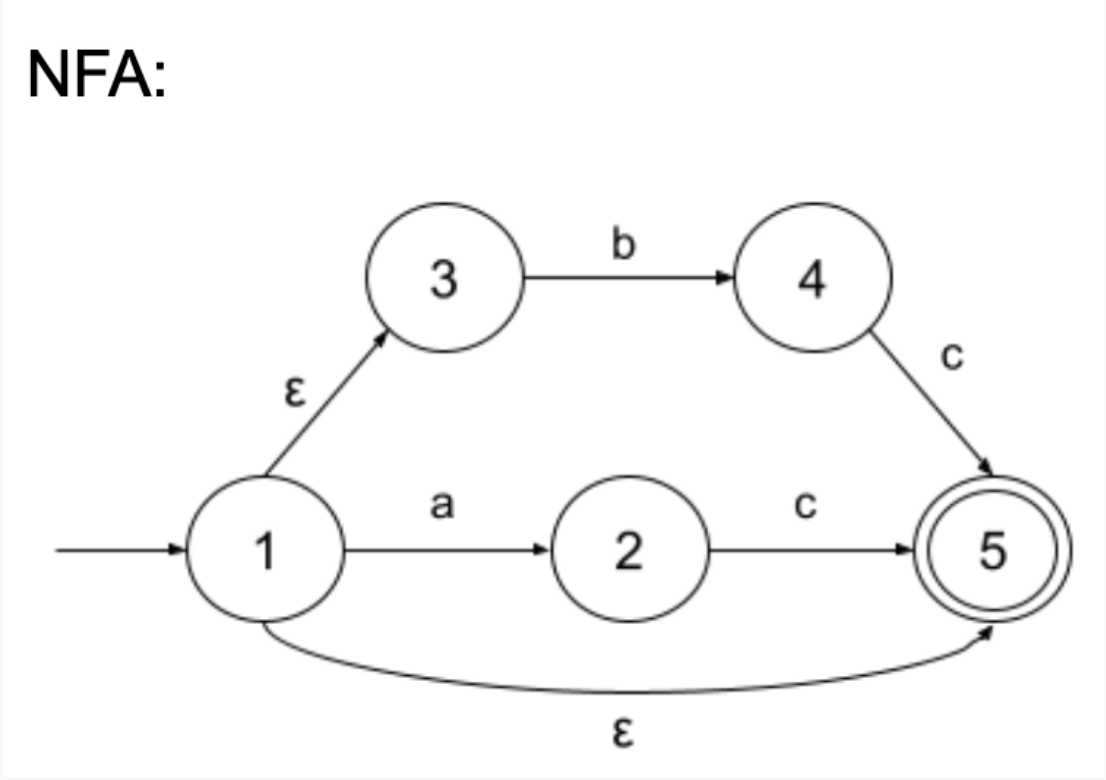
- [ ] X
- [ ] Y
- [ ] Z

Save Answer

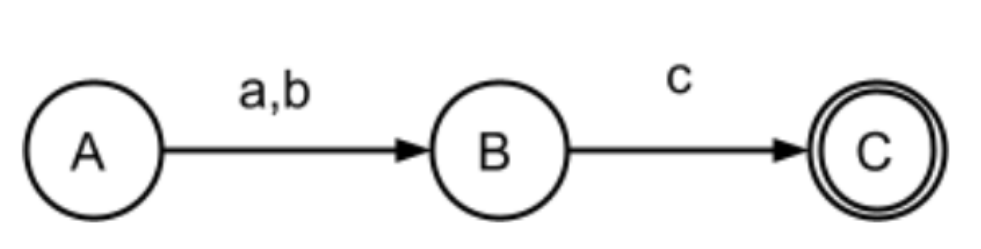
Q2.8 FA fixup

3 Points

The following NFA and DFA do not accept the same strings. How can we change the DFA so that they do (while keeping it as a DFA)?



DFA:



(Note that these FAs are independent; we are not asking you to convert one to the other.)

- ○ Add an epsilon transition from state A to state C.
- ○ Add a 4th state to the DFA. Have state A transition to this new state on 'b', and have that state transition to state C on 'c'.
- ○ Make state A a final state.
- ○ There is no way to make them equivalent.

Save Answer

Q3 Context Free Grammars

13 Points

In the problems below, capital letters represent nonterminals (e.g., A,B,C) and lowercase letters represent terminals (e.g., a,b,c).

Q3.1

5 Points

Which of the following strings will be accepted by the CFG below (A is the start state)

$A \rightarrow wA \mid xB \mid \varepsilon$
$B \rightarrow xC \mid C$
$C \rightarrow zC \mid z$

- ☐ w
- ☐ xxxw
- ☐ wwwx
- ☐ empty string
- ☐ xxzzz

Save Answer

Q3.2

5 Points

Write a CFG for palindromes over alphabet $\Sigma = \{\mathbf{e}, \mathbf{f}, \mathbf{g}\}$. (Note: the empty string is a palindrome.)

Enter your answer here

Save Answer

Q3.3
3 Points

Which one of the following CFGs generates exactly the set of strings $\mathbf{a}^n\mathbf{b}^n\mathbf{c}^n$?

- S → abSc | ε
- S → aSSc | T T → bT | ε
- S → abcS | ε
- None of the above

Save Answer

Q4 Parsing
21 Points

The following questions relate to the process of constructing a predictive parser from a CFG.

Q4.1 FIRST sets
5 Points

Given the grammar below, which terminals are contained in the FIRST set of S (check all that apply)?
S → ABC
A → a | b | ε
B → c | ε
C → d | ε

- [ ] a
- [ ] b
- [ ] c
- [ ] d
- [ ] ε

Save Answer

Q4.2 Left factoring

5 Points

A predictive parser requires FIRST sets of the right-hand sides of each non-terminal's productions to be non-overlapping. Change the following grammar so that this is the case. (Hint: use the left-factoring algorithm.)

S ⟶ ab | aT | T
T ⟶ bc | b

Enter your answer here

Save Answer

Q4.3 Right recursion

5 Points

A predictive parser does not work on left-recursive CFGs, since they would induce infinite recursion. Fix this CFG so that it is right-recursive, intead.

S ⟶ SaS | V
V ⟶ a | b | c

Enter your answer here

Save Answer

Q4.4 Construct the CFG from the parser

6 Points

The following is parsing code in the style given in the **lecture slides**, and which you (mostly) followed when implementing project p4a. Reading the code, determine the CFG that it is parsing.

```
let tok_list:(char list) ref = ref ...;; (* lexing tokens stored here *)
exception ParseError of string

let lookahead () =
  match !tok_list with
  | [] -> None
  | (h::t) -> Some h

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

let rec parse_S () =
    match lookahead () with
    | Some 'a' ->
```

```
            match_tok 'a';
            parse_S ();
            match_tok 'a'
    |_-> parse_T ()

and parse_T () =
    match lookahead () with
    | Some 'b' ->
            match_tok 'b';
            parse_S ();
            match_tok 'b'
    | Some 'c' ->
            match_tok 'c'
    | _ -> raise (ParseError("Bad Input"))
```

Enter your answer here

Save Answer

Q5 Operational Semantics

10 Points

These questions derive from material in the **lecture on operational semantics**.

Q5.1

0 Points

A *definitional interpreter* is

- ○ An interpreter for a language's definitions
- ○ An executable version of the language's operational semantics
- ○ An interpreter written with a context-free grammar
- ○ An operational semantics for interpreted languages

Save Answer

Q5.2

2 Points

An *environment* `A` in the operational semantics judgment, `A; e ⇒ v`, is a partial map from variables `x` to values `v`. What do we write to extend an existing environment `A` with a new mapping from variable `y` to value `3`, overriding any previous mapping for `y`?

- ○ `y:3,A`
- ○ `A;y:3`
- ○ `A,y:3`
- ○ `A{y/3}`

Save Answer

Q5.3

2 Points

An operational semantics is often defined using *rules of inference*, a notation taken from formal logic. What is the term for an inference rule with no hypotheses?

- A conjecture
- A judgment
- A derivation
- An axiom

Save Answer

Q5.4 Extending MicroCaml

6 Points

Recall MicroCaml from the lecture. Imagine we want to extend this language with support for pairs. The grammar for abstract syntax trees (ASTs) would be extended as follows, where `v` represents *values*, `x` represents *variables*, and `e` represents *expressions*.

```
v ::= ... | (v,v)
e ::= ... | (e,e) | let (x,x) = e in e
```

Thus a pair value is something like `(1,1)` or `(true,false)` (since `1`, `true`, and `false` are, themselves, values). The contents of pairs are accessed by pattern matching, per the added `let` syntax. Here are three example judgments that should hold:

```
A; (2+3,false)                              ⇒  A; (5,false)
A,y:5; let x = (2+3,y) in x                 ⇒  A,y:5; (5,5)
A; let x = (1,2) in let (y,z) = x in  y+z   ⇒  A; 3
```

Complete the operational semantics rules given below, by filling in the blanks to complete the missing or partial judgments (add as much math as you need to).

```
A; e1 ⇒ v1    ________#1
-----------------------------------------
A; (e1,e2) ⇒ (v1,v2)


A; e1 ⇒ (v1,v2)    A,x1:v1,________#2
-----------------------------------------
A; let (x1,x2) = e1 in e2   ⇒   v
```

#1

Enter your answer here

#2

Enter your answer here

Save Answer

Q6 Security

11 Points

The following questions relate to lectures on software and web security.

Q6.1

3 Points

Programming languages commonly associated with buffer overflows include which of the following (check all that apply) ?

- [ ] C
- [ ] Java
- [ ] Ruby

Save Answer

Q6.2

2 Points

True or false: It is safer for a program to only accept good inputs rather than try to reject or sanitize bad ones.

- ○ True
- ○ False

Save Answer

Q6.3 Security

2 Points

Consider the following Ruby function which asks the user for an IP address and then runs the system's `ping` on the address:

```
def ping_host
    puts "Please enter an IP address:"
    addr = gets
    system "ping #{addr}"
end
```

To which kind of attack is this code susceptible?

- SQL injection
- Command injection
- XSS
- Buffer overflow

Save Answer

Q6.4

4 Points

The following Ruby code is vulnerable to SQL injection:

```ruby
def authenticate(username, password)
    if username =~ /[^A-Za-z0-9]/ then
        puts "invalid chars in username"
        return nil
    end

    user_info = db.execute "SELECT * FROM users
        WHERE (username='#{username}') AND (password='#{password}')"

    return user_info
end
```

What is the reason for the vulnerability? Refer to specific code in your answer.

Enter your answer here

How would you change the code to avoid the vulnerability? Refer to the specific code you would change, and your approach.

Enter your answer here

Save Answer

Save All Answers

Submit & View Submission ›