Exam 2 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

- Programming Language Concepts [13pts]
- Finite Automata [16pts]
- Context-Free Grammars [14pts]
- Parsing [18pts]
- Operational Semantics [13pts]
- Evaluation [10pts]
- Lambda Calculus [16pts]

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 75 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

13 Points

Q2.1

3 Points

Can any CFG be converted to an equivalent NFA? If not, provide a counterexample.

Enter your answer here

Save Answer

Q2.2

2 Points

An NFA is more powerful than a DFA; that is, some NFA represents a language that no DFA can represent.

- [ ] True
- [ ] False

Save Answer

Q2.3

3 Points

Which of the following can be used to express the language consisting of palindromes over the alphabet of lowercase English letters (a-z)?

- [ ] Regular expression
- [ ] NFA
- [ ] DFA
- [ ] CFG

Save Answer

Q2.4
2 Points

A parser is a program which takes in an abstract syntax tree (AST) and produces a result value.

- [ ] True
- [ ] False

Save Answer

Q2.5
3 Points

When we're interested in learning a language, it's often easier to "learn by example" rather than to look up the operational semantics for the language. What is a benefit of a language developer including operational semantics in their newly-designed programming language (as opposed to just providing several examples)?

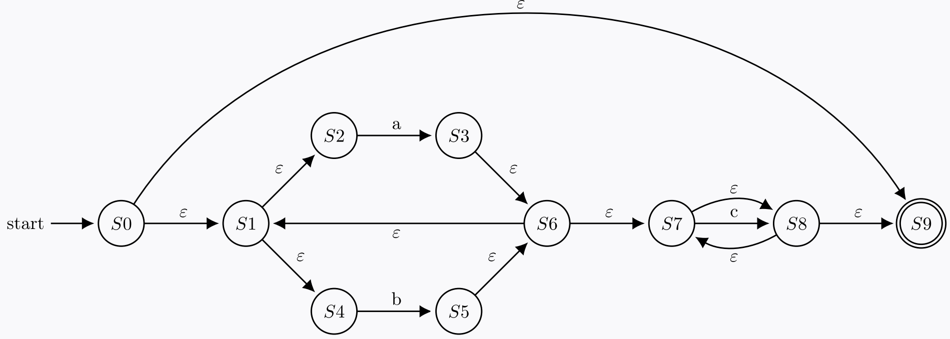
Enter your answer here

Save Answer

Q3 Finite Automata
16 Points

Q3.1
4 Points

Consider the following NFA:



Which of the following regular expressions is equivalent to this NFA?

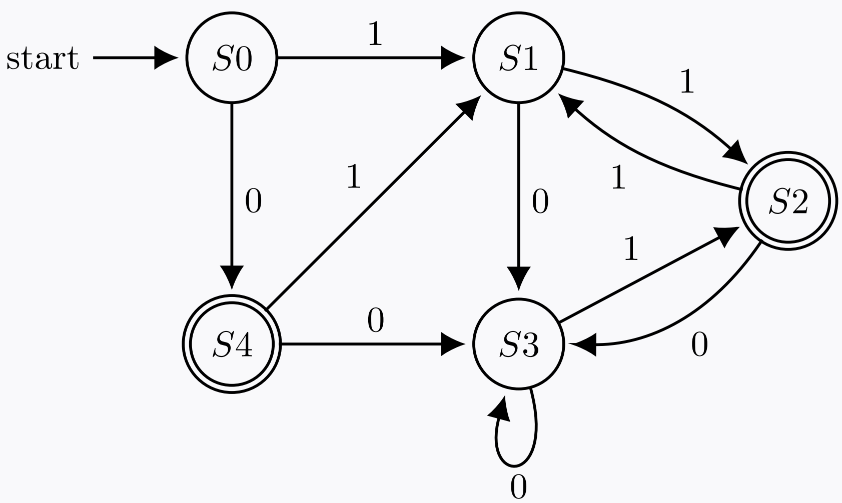
- ( ) `((a|b)+c*)?`
- ( ) `((a|b)*c*)+`
- ( ) `((a|b)+c?)+`
- ( ) `((a|b)*c?)?`

Save Answer

Q3.2

4 Points

Consider the following DFA:



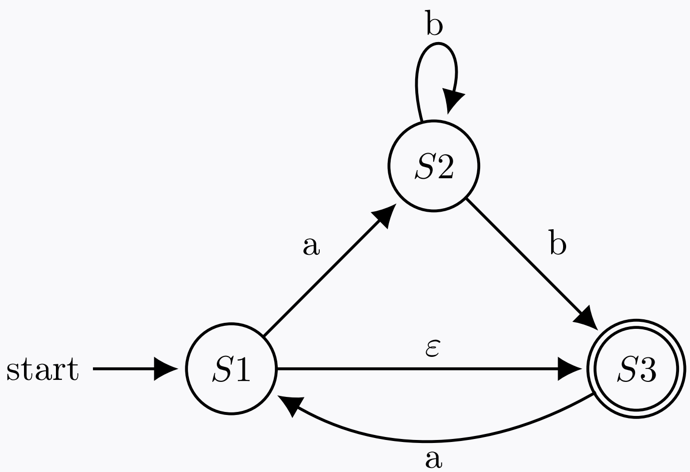
Which strings are accepted by this DFA? Select all that apply

- [ ] 11101
- [ ] 00101101
- [ ] 01110110
- [ ] 111101
- [ ] 1100

Save Answer

Q3.3
8 Points

Consider the following NFA:



Convert this to a DFA. Write your answer as an OCaml type from project 3:

```
type ('q, 's) transition = 'q * 's option * 'q
type ('q, 's) nfa_t = {
  sigma: 's list;
  qs: 'q list;
  q0: 'q;
  fs: 'q list;
  delta: ('q, 's) transition list;
}
```

You do not need to show all the steps, only the final answer.

Enter your answer here

Save Answer

Q4 Context-Free Grammars
14 Points

Q4.1
5 Points

Write a CFG that generates strings of the form $a^x b c^y$ with $x \geq y \geq 0$. You can use the ε symbol or just write "epsilon" where needed.

Enter your answer here

Save Answer

Q4.2

4 Points

Which strings can be generated by the following CFG with start symbol X?

$$X \rightarrow aaXb \mid cY \mid \varepsilon$$
$$Y \rightarrow acY \mid cY \mid Z$$
$$Z \rightarrow bZ \mid b$$

- [ ] *(empty string)*
- [ ] bbb
- [ ] aaacb
- [ ] cacbb

Save Answer

Q4.3

5 Points

Write a regular expression that accepts the same strings as the following CFG with start symbol W:

$$W \rightarrow m \mid X$$
$$X \rightarrow Y \mid nX$$
$$Y \rightarrow pY \mid pZ$$
$$Z \rightarrow qZ \mid \varepsilon$$

Enter your answer here

Save Answer

Q5 Parsing

18 Points

Q5.1

10 Points

Consider the following context-free grammar:

$$S \rightarrow aS \mid bS \mid cTb$$
$$T \rightarrow aT \mid bT \mid c$$

Write the OCaml functions to parse this grammar from a list of tokens (all of which are strings). You are given the following two functions to help you:

```
let lookahead toks =
    match toks with
    | h::t -> Some h
    | [] -> None

let match_tok toks x =
    match toks with
    | h::t when h = x -> t
    | [] -> raise (ParseError "bad match")

let rec parse_S toks =
    (* TODO *)

let rec parse_T toks =
    (* TODO *)
```

Note that you do not need to produce an AST, just parse the input, failing if it is invalid. The parse functions should return the updated token lists.

Implement the `parse_S` function below:

Enter your answer here

Implement the `parse_T` function below:

Enter your answer here

Save Answer

Q5.2

4 Points

Can the following grammar be parsed by a recursive descent parser as it is written? If not, briefly explain why. If it can, no explanation is necessary.

$$S \to Sa \mid Sb \mid T$$
$$T \to a \mid b \mid c$$

Enter your answer here

Save Answer

Q5.3

4 Points

What is the role of a tokenizer/lexer?

Enter your answer here

What is the role of a parser?

Enter your answer here

Save Answer

Q6 Operational Semantics

13 Points

Consider the following operational semantics rules:

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

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

$$\frac{A; e_1 \to \textbf{true} \qquad A; e_2 \to v}{A;\ \textbf{if}\ e_1\ \textbf{then}\ e_2\ \textbf{else}\ e_3 \to v} \qquad \frac{A; e_1 \to \textbf{false} \qquad A; e_3 \to v}{A;\ \textbf{if}\ e_1\ \textbf{then}\ e_2\ \textbf{else}\ e_3 \to v}$$

Fill in the blanks in the proof below:

$$\frac{\dfrac{\dfrac{\boxed{1}}{A, x : 3; x \to 3} \qquad A, x : 3; \boxed{2} \qquad \boxed{3}}{A, \boxed{4}\ ; x > 2 \to \textbf{true}} \qquad A; x : 3; \boxed{5}}{A, \boxed{6}\ ; \textbf{if}\ \boxed{7}\ \textbf{then}\ 151\ \textbf{else}\ 251 \to \boxed{8}}$$

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

Blank 7:

Enter your answer here

Blank 8:

Enter your answer here

Save Answer

Q7 Evaluation

10 Points

Consider the following rules which describe basic arithmetic operations on integers:

$$\textsc{Int}\ \frac{}{n \to n} \qquad\qquad \textsc{Add}\ \frac{e_1 \to n_1 \qquad e_2 \to n_2 \qquad n_3 = n_1 + n_2}{e_1 + e_2 \to n_3}$$

$$\textsc{Mul}\ \frac{e_1 \to n_1 \qquad e_2 \to n_2 \qquad n_3 = n_1 \cdot n_2}{e_1 \cdot e_2 \to n_3}$$

and also the following AST type:

```
type exp =
| Int i of int
| Add of exp * exp      (* exp1 + exp2 *)
| Mul of exp * exp      (* exp1 * exp2 *)

let rec eval t =
    (* TODO *)
```

Write the `eval` function which takes as an argument an AST of the type above, and returns the resulting value.

Implement the `eval` function below:

Enter your answer here

Save Answer

Q8 Lambda Calculus
16 Points

Q8.1
4 Points

Consider the following beta-reduction which contains an error:

$(\lambda x.x\ (\lambda y.y\ x))\ y\ x$
$y\ (\lambda y.y\ y)\ x$

Identify the error, and describe why it is incorrect.

Enter your answer here

Save Answer

Q8.2
6 Points

Consider the following lambda expression:
$(\lambda x.y\ z)\ ((\lambda x.x\ x\ y)\ (\lambda z.z\ z\ y))$

Show a sequence of reductions which result in beta-normal form. Show each step of the reduction.

Enter your answer here

It's also possible to reduce this expression infinitely without ever reaching beta-normal form. Show at least two steps of such a reduction.

Enter your answer here

Save Answer

Q8.3
6 Points

Given the following lambda calculus encodings:

$\mathbf{0} = \lambda f.\lambda y.y$
$\mathbf{1} = \lambda f.\lambda y.f\ y$
$\mathbf{2} = \lambda f.\lambda y.f\ (f\ y)$
$\mathbf{true} = \lambda x.\lambda y.x$
$\mathbf{false} = \lambda x.\lambda y.y$
$\mathbf{iszero} = \lambda z.z\ (\lambda y.\mathbf{false})\ \mathbf{true}$

Prove that $\mathbf{iszero}\ 1 = \mathbf{false}$. Be sure to show all steps for full credit.

Enter your answer here

Save Answer

Save All Answers

Submit & View Submission