Midterm 2 from Fall 2021

STUDENT NAME

Search students by name or email...

Q1 Instruction

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 can also use `irb` and `utop`. 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

- DFA and NFA
- NFA to DFA
- Context-Free Grammars
- Parsing
- Operational Semantics
- Lambda Calculus
- Rust

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. 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

Save Answer

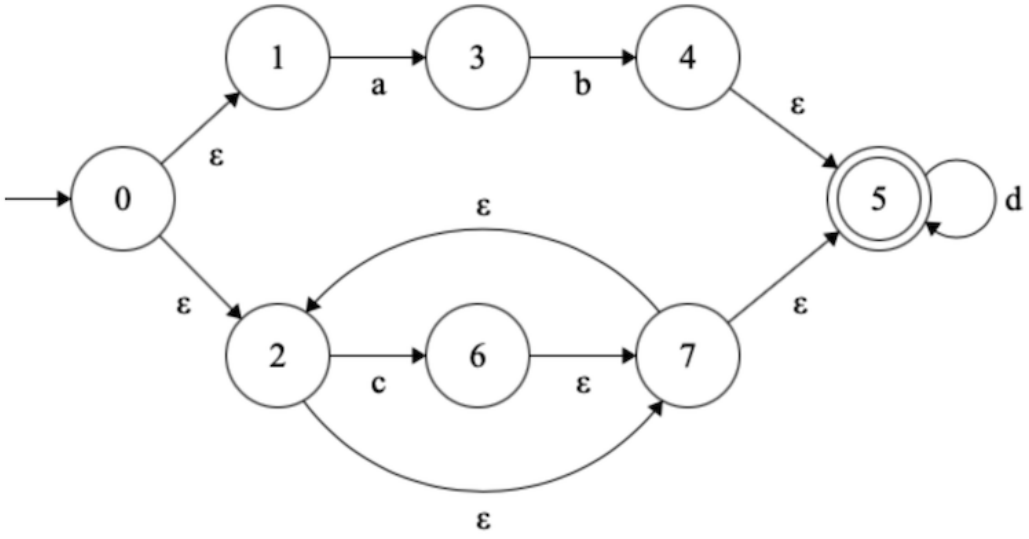
Q2 DFA and NFA

10 Points

Q2.1 DFA and NFA

4 Points

Which strings will this NFA accept?



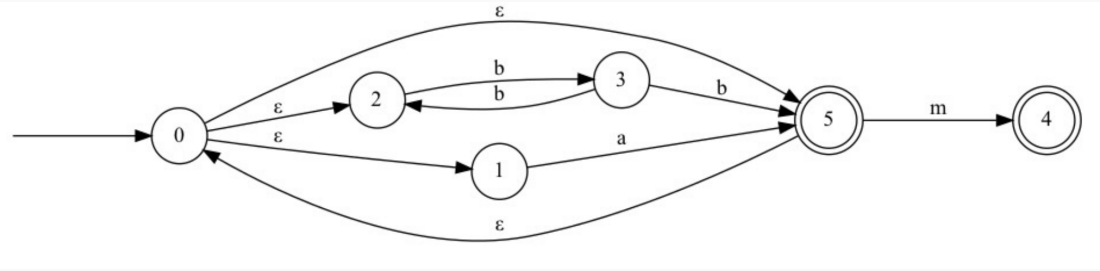
- [ ] ab
- [ ] cccccab
- [ ] ddddd
- [ ] ad

Save Answer

Q2.2 NFA to RegEx

6 Points

What regular expression does this NFA correspond to?



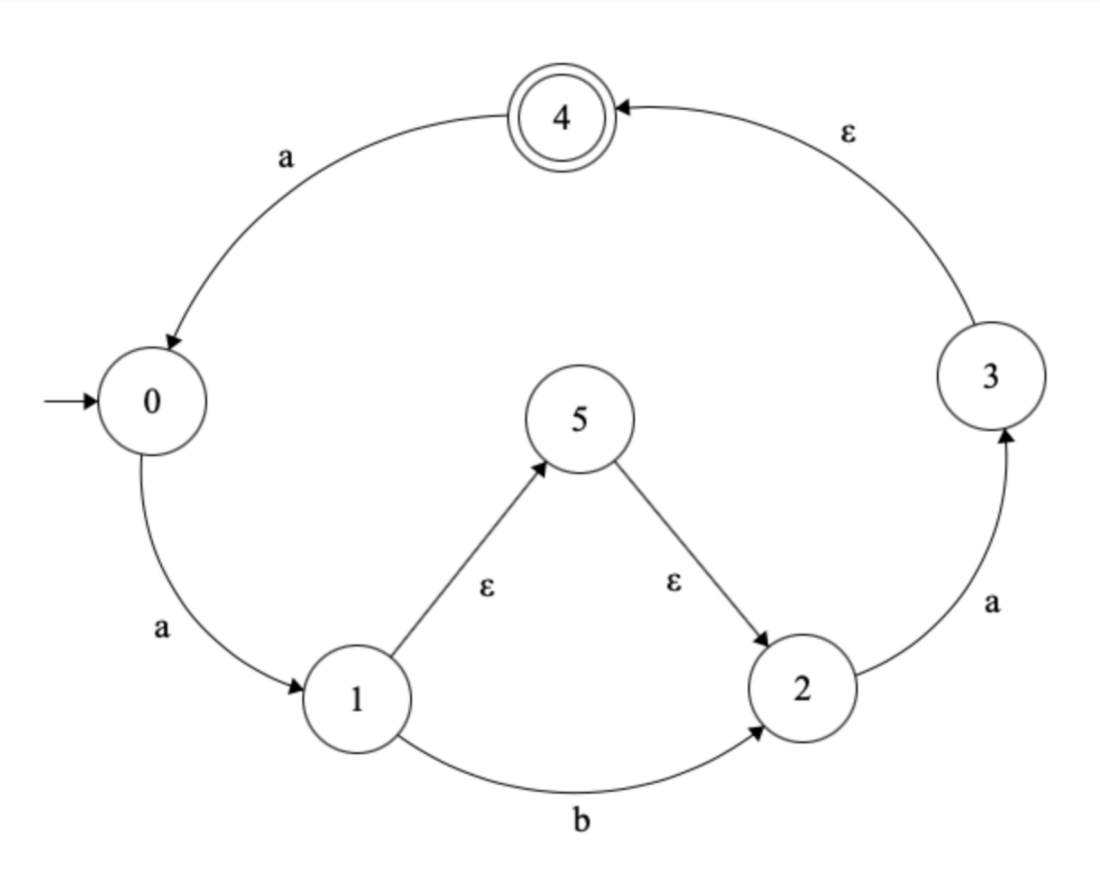
Enter your answer here

Save Answer

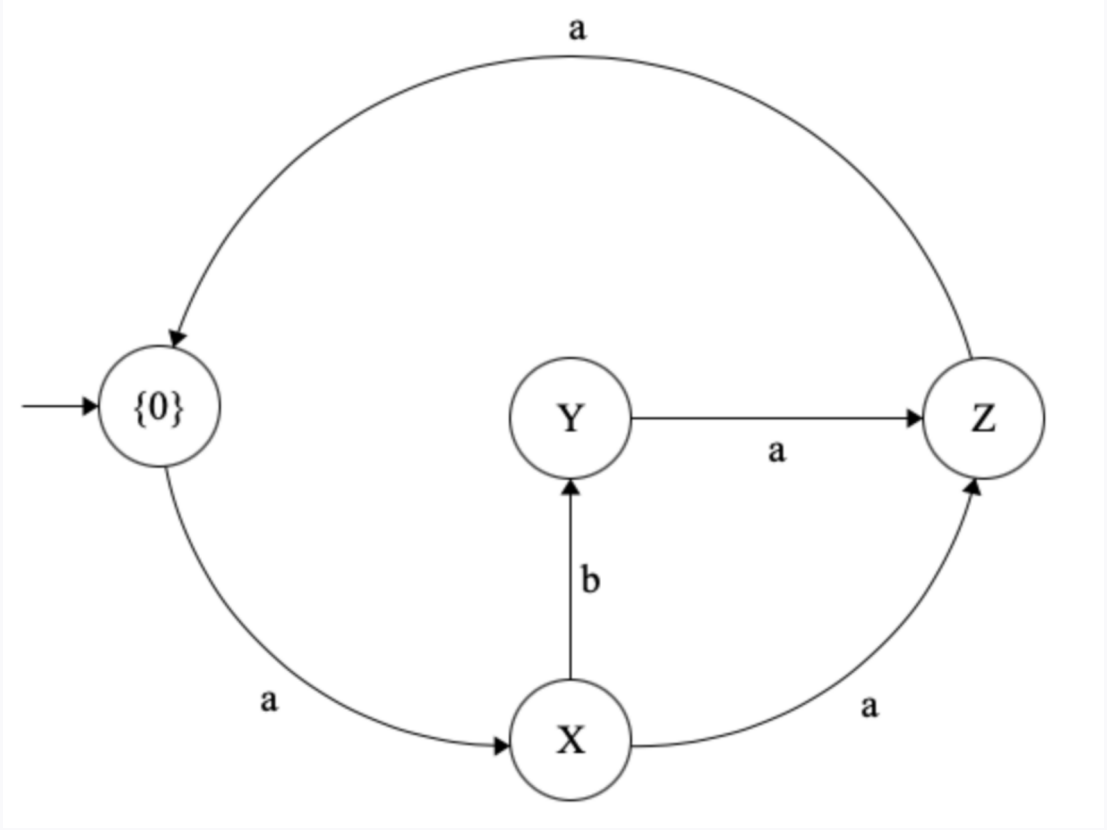
Q3 NFA to DFA

16 Points

Consider the following NFA:



Using subset construction, the above NFA can be converted to the DFA.



Q3.1 NFA to DFA 1

4 Points

In this DFA, which states from the original NFA make up the state X?

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4
- [ ] 5

Save Answer

Q3.2 NFA to DFA 2

4 Points

In this DFA, which states from the original NFA make up the state Y?

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4
- [ ] 5

Save Answer

Q3.3 NFA to DFA 3

4 Points

In this DFA, which states from the original NFA make up the state Z?

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4
- [ ] 5

Save Answer

Q3.4 NFA to DFA 4

4 Points

In this DFA, which states are final?

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

Save Answer

Q4 Context-Free Grammars

18 Points

Q4.1 CFG

6 Points

Write a CFG over the alphabet $\Sigma = \{0, 1\}$ that recognizes strings that start with a `1`, end with a `0`, and have any number of `0`s or `1`s in between.

Enter your answer here

Save Answer

Q4.2 Left Recursion vs. Right Recursion

6 Points

Consider the following CFG

```
S -> S + T | T
T -> P * T | P
P -> 1 | 2 | 3 | (S)
```

Notice that this grammar is left recursive, write down every rule (with or without any changes) so that it is right recursive.

Hint: Write the new first rule on the first line, the new second rule on the second line, etc.

Enter your answer here

Save Answer

Q4.3 FIRST Set

6 Points

Find the FIRST sets for each non-terminal in the following grammar:

```
S -> TU | Ua
T -> da | Ub
U -> g | ε
```

FIRST(S)=

Enter your answer here

FIRST(T)=

Enter your answer here

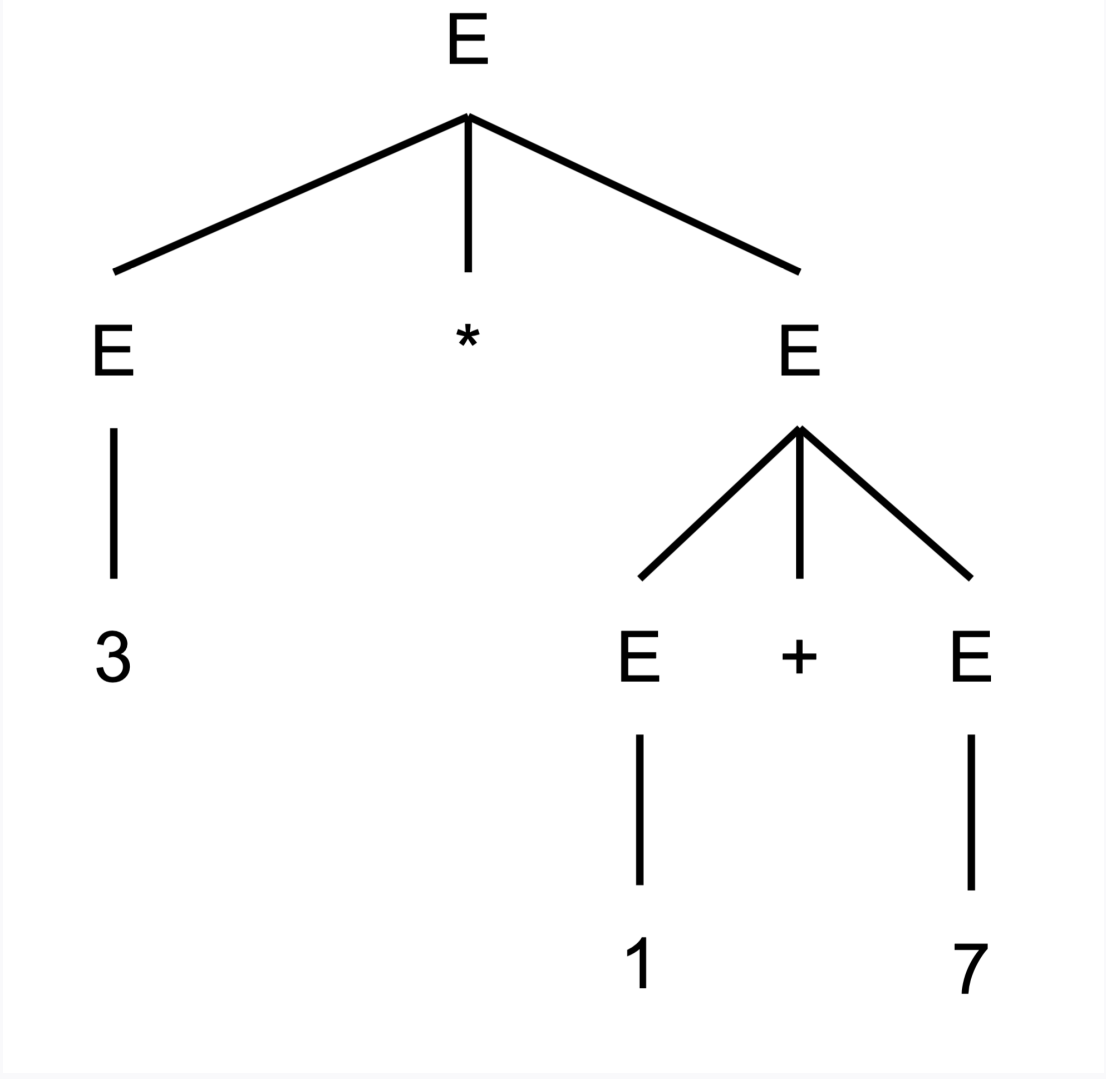
FIRST(U)=

Enter your answer here

Save Answer

Q5 Parsing
18 Points

Q5.1 Parse Tree
6 Points



Write an `exp` value to describe the corresponding AST of the above parse tree.

```
type exp =
    | Num of int
    | Plus of exp * exp
    | Mul of exp * exp
```

Enter your answer here

Save Answer

Q5.2 Recursive Descent Parser

12 Points

You will implement a recursive descent parser for the following CFG:

```
S -> TU | a
T -> da | Ub
U -> g
```

You should use these functions and definitions:

```
type token =
    | Tok_a
    | Tok_b
    | Tok_d
    | Tok_g

(* Note that these are imperative implementations.
   You may assume that `tok_list` has been filled by a lexer. *)

let tok_list = ref []

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

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

Your functions should return the unit value `()` if they successfully parse, otherwise they should `raise (ParseError "message")`. (The contents of the message string do not matter.)

Enter your code for the functions below:

```
let rec parse_S () =
```

Enter your answer here

```
and parse_T () =
```

Enter your answer here

```
and parse_U () =
```

Enter your answer here

Save Answer

Q6 Operational Semantics

13 Points

Consider the following grammar of expressions in a new programming language:

$$
\begin{array}{rcl}
e & ::= & x \\
& | & 1 \\
& | & 0 \\
& | & x \texttt{ = } e \texttt{ in } e \\
& | & \blacksquare\, e \\
& | & e \bullet e \\
& | & (e)
\end{array}
\qquad\qquad
\begin{array}{rcl}
v & ::= & 1 \\
& | & 0
\end{array}
$$

(The x represents variable names. We use e to mean *expressions* and v to mean *values*. Parentheses are only used to prevent ambiguity, e.g., the term $(\blacksquare\, e) \bullet e$ is distinct from the term $\blacksquare\, (e \bullet e)$, but the parens have no other purpose.)

Answer the questions about this language with the following operational semantics. Note that each semantic rule is numbered for reference in your answers.

$$\frac{}{A;\ 1 \Rightarrow 1}\ (1) \qquad \frac{}{A;\ 0 \Rightarrow 0}\ (2) \qquad \frac{A(x) = v}{A;\ x \Rightarrow v}\ (3)$$

$$\frac{A;\ e_1 \Rightarrow v_1 \qquad v_2 \text{ is } 0 \text{ if } v_1 \text{ is } 1, \text{ otherwise } v_2 \text{ is } 1}{A;\ \blacksquare\, e_1 \Rightarrow v_2}\ (4)$$

$$\frac{A;\ e_1 \Rightarrow v_1 \qquad A;\ e_2 \Rightarrow v_2 \qquad v_3 \text{ is } 0 \text{ if } v_1 = v_2, \text{ otherwise } v_3 \text{ is } 1}{A;\ e_1 \bullet e_2 \Rightarrow v_3}\ (5)$$

$$\frac{A;\ e_1 \Rightarrow v_1 \qquad A, x : v_1;\ e_2 \Rightarrow v_2}{A;\ x \texttt{ = } e_1 \texttt{ in } e_2 \Rightarrow v_2}\ (6)$$

Q6.1 Operational Semantics 1

3 Points

Rules 1 and 2 (but none of the others) are examples of...

- Axioms
- Semantics
- Environments
- Operations

Save Answer

Q6.2 Operational Semantics 2

3 Points

The semantics given are...

- ○ Small-step
- ○ Big-step

Save Answer

Q6.3 Operational Semantics 3

3 Points

The "A" used in the operational semantics is called a(n)

- ○ Expression
- ○ Term
- ○ Environment
- ○ Alphabet

Save Answer

Q6.4 Operational Semantics 4

4 Points

(Read the full text of this question carefully!)

Using the given operational semantics, you must evaluate the following expression:

$$x \texttt{ = } 0 \texttt{ in } 1 \bullet (\blacksquare\ x)$$

First, fill in the holes in the proof tree shown below by writing the number of the rule for each hole. Then, tell us what the expression evaluates to (the $?$ in the bottom judgment).

ONLY WRITE THE RULE'S NUMBER, DO NOT WRITE THE RULE'S TEXT.

$$\dfrac{\dfrac{}{}\,(b) \qquad \dfrac{\dfrac{}{}\,(d) \qquad \dfrac{\dfrac{}{}\,(f) \qquad \dfrac{}{}\,(e)}{A, x : 0;\ 1 \bullet (\blacksquare\ x) \Rightarrow\ ?}\,(c)}{A;\ x \texttt{ = } 0 \texttt{ in } 1 \bullet (\blacksquare\ x) \Rightarrow\ ?\,(g)}\,(a)$$

Hints

1. Each rule is used exactly once.
2. Proof trees are filled from the bottom up and from left to right.
3. The result in part (g) should be a *value*. There are only two possible values in this language!

We have copied the semantics from the top of question 6 here so you don't have to scroll so much:

$$\frac{}{A;\ 1 \Rightarrow 1}\ (1) \qquad \frac{}{A;\ 0 \Rightarrow 0}\ (2) \qquad \frac{A(x) = v}{A;\ x \Rightarrow v}\ (3)$$

$$\frac{A;\ e_1 \Rightarrow v_1 \qquad v_2 \text{ is 0 if } v_1 \text{ is 1, otherwise } v_2 \text{ is 1}}{A;\ \blacksquare\ e_1 \Rightarrow v_2}\ (4)$$

$$\frac{A;\ e_1 \Rightarrow v_1 \qquad A;\ e_2 \Rightarrow v_2 \qquad v_3 \text{ is 0 if } v_1 = v_2, \text{ otherwise } v_3 \text{ is 1}}{A;\ e_1\ \bullet\ e_2 \Rightarrow v_3}\ (5)$$

$$\frac{A;\ e_1 \Rightarrow v_1 \qquad A, x : v_1;\ e_2 \Rightarrow v_2}{A;\ x\ \texttt{=}\ e_1\ \texttt{in}\ e_2 \Rightarrow v_2}\ (6)$$

(a)

Enter your answer here

(b)

Enter your answer here

(c)

Enter your answer here

(d)

Enter your answer here

(e)

Enter your answer here

(f)

Enter your answer here

(g) (Result)

Enter your answer here

Save Answer

Q7 Lambda Calculus

15 Points

In your answers for this section, you may write the lambda symbol as `λ`, `\`, or `L`, but please be consistent!

Q7.1 Lambda Calculus 1

4 Points

Reduce the expression as far as possible (Show your work for partial credits):

```
(λx. λx. y x) (λx. y x) x
```

Enter your answer here

Save Answer

Q7.2 Lambda Calculus 2

6 Points

Perform an α-conversion to the following expression:

```
(λy. λz. y z) (λa. y) (x)
```

Enter your answer here

Then, apply as many β-reductions as possible to it **without performing any other α-conversions**.

Enter your answer here

Save Answer

Q7.3 Lambda Calculus 3

5 Points

Given the following definitions:

```
true  = λx. λy. x
false = λx. λy. y
and   = λx. λy. x y false
```

Prove that `and true false` is equivalent to `false`. (Show all steps involving β-reduction and substitution/replacement for full credit.)

Enter your answer here

Save Answer

Q8 Rust
10 Points

Q8.1 Rust: Ownership
4 Points

```
let a = String::from("cmsc330");
let b = a;
let c = &b;
let d = c;
```

Which variable is the owner of the string `"cmsc330"`?

- a
- b
- c
- d

Save Answer

Q8.2 Rust: Fill in the blank
6 Points

Consider the following Rust code:

```
fn foo(s: _____) {   // #1
  let mut i = s.len();
  while i > 0 {
    println!("{}", &s___); //#2
    i = i - 1;
  }
}

fn main() {
  let s = String::from("CMSC330");
  foo(_____); //#3
}
```

Fill in the 3 blanks such that the program, when run, outputs:

```
0
30
330
C330
SC330
MSC330
```

#1

Enter your answer here

#2

Enter your answer here

#3

Enter your answer here

Save Answer

Save All Answers

Submit & View Submission >