Final Exam from Spring 2021 (Practice)

STUDENT NAME

Search students by name or email...

Q1 PL Concepts

9 Points

Q1.1 Scoping

2 Points

Consider the following OCaml code:

```
let word = "Hello!";;
let foo word = word;;
let bar () =
  let word = "Goodbye!" in
  foo word;;
print_string (bar());;
```

What would OCaml print if it used *static* (aka *lexical*) scoping?

- "Hello!"
- "Goodbye!"

What would OCaml print if it used *dynamic* scoping?

- "Hello!"
- "Goodbye!"

Save Answer

Q1.2 Program execution

1 Point

Which statement is most true about a compiler?

- It runs a program directly
- It translates a program to machine code
- It translates a program from one language to another
- It is not applicable to dynamically typed languages

Save Answer

Q1.3 Dynamic typing

1 Point

Which of the following languages is dynamically typed?

- ○ OCaml
- ○ Ruby
- ○ C
- ○ Rust
- ○ Java

Save Answer

Q1.4 Automatic memory management

2 Points

Reference counting has which of the following benefits over tracing garbage collection?

- ☐ It performs compaction to reduce fragmentation
- ☐ It determines reachability incrementally, as the program executes
- ☐ Memory can be freed immediately once it becomes unreachable
- ☐ Cyclic garbage is not retained

Save Answer

Q1.5 Traits and interfaces

1 Point

What is a difference between Rust *traits* from Java *interfaces*?

- ○ A Java class can only have one interface but a Rust type can have multiple traits
- ○ A Java interface only applies to classes that explicitly declare it, but a Rust trait can be applied to any type
- ○ A Java interface supports generic type parameters, but Rust traits do not

Save Answer

Q1.6 Property Based Testing

2 Points

```
let test_mystery_pbt =
  Test.make
  ~name:"?"
  ~count:1000
  (list small_int)
  (fun x ->
```

```
    List.length (uniq x) = List.length (uniq (List.rev x))
  )
```

What property is this code testing?

- ○ `uniq` does not change the length of the list
- ○ `List.length` returns an error on a reversed list
- ○ `uniq` should return a list of the same length whether given a list or its reverse
- ○ `uniq` should return the same result whether given a list or its reverse

What sort of input is the above test generating?

- ○ List of small strings
- ○ List of small integers
- ○ An integer
- ○ A string

Save Answer

Q2 Ruby

16 Points

Q2.1 Code blocks

3 Points

Consider the following code:

```ruby
def extra_credit(grade)
    if block_given?
        _______      # 1
    else
        grade
    end
end
puts extra_credit(45) { |x| x + 20 }
puts extra_credit(60)
puts extra_credit(95) { |x| _______ }       # 2
```

Fill in the blanks above in order to create the following output:

```
65
60
0
```

#1

Enter your answer here

#2

Enter your answer here

Save Answer

Q2.2 Hashes

2 Points

Consider the following code:

```
x = Hash.new []

_____________

puts x["3xx"].inspect
puts x["5xx"].inspect
```

Fill in the blank to create the following output:

```
[330, 351]
[]
```

Enter your answer here

Save Answer

Q2.3 Regular Expressions

3 Points

Write a regular expression in Ruby that accepts a string if and only if the following criteria are met:

1. The string starts with the letter "a".
2. It has length 4.
3. It ends with the letter "z".
4. It has any letter (upper or lower case) or digit in the other positions.

When a string is matched against the regular expression, back reference $1 should extract the middle two characters.

Enter your answer here

Save Answer

Q2.4 Arrays

4 Points

Consider the following code:

```
numbers = [8,4,2,1]
output = numbers._______ { |val|    # 1
    __________                      # 2
}
puts output.inspect
```

Fill in the blanks so as to print the output:

```
[4, 2, 1, 0]
```

#1

Enter your answer here

#2

Enter your answer here

Save Answer

Q2.5 Classes

4 Points

Consider the following code:

```
class Comp
  ____________________      # 1
  def initialize(n,m)
    @n = n
    @m = m
  end
  def +(c)
    Comp.new(@n+c.n, @m+c.m)
  end
  def to_s
    _________________       # 2
  end
end

x = Comp.new(1,2)
y = Comp.new(3,4)
z = x+y
puts "x = #{x}; y = #{y}; z = #{z}"
```

Fill in the blanks so as to print the output:

```
x = 1 + 2i; y = 3 + 4i; z = 4 + 6i
```

#1

Enter your answer here

#2

Enter your answer here

Save Answer

Q3 OCaml
16 Points

Q3.1 Typing
2 Points

Write an expression with the following type:

```
'a -> int
```

Enter your answer here

Save Answer

Q3.2 Typing
3 Points

Write an expression with the following type:

```
'a -> ('a -> 'b) list -> 'b list
```

Enter your answer here

Save Answer

Q3.3 Typing
2 Points

Write the type of the following expression, or explain why there is a type error.

```
fun x y z -> if x = y then
               y + z
             else if z then
               y
             else
               x
```

Enter your answer here

Save Answer

Q3.4 What's the missing code?

2 Points

What would you put in the blank so that the following code returns `13` ?

```
List.fold_left (fun a x -> a + _________) 0 [6; 0; 4]
```

Enter your answer here

Save Answer

Q3.5 What's the input?

2 Points

What would you put in the blank so the following code returns `[Some 3; None; Some 2]`?

```
let rec f l a =
  match l with
  | [] -> a
  | h::t -> let z = if h = 0 then None
                    else Some (12 / h) in
            f t (z::a) in
f _______________ []
```

Enter your answer here

Save Answer

Q3.6 Coding

5 Points

Write a function that accepts a list of lists of integers and returns a list of integers representing the smallest number in each sublist. Example:

```
smallest [[20;0]; [2;-10;3]] = [0; -10]
smallest [[9;1]; [100]; [20; 99]] = [1; 100; 20]
smallest [] = []
smallest [[];[]] = [0;0]
```

Complete the code below. You may assume you have a function `min` such that `min x y` returns whichever is smallest of `x` and `y`.

```
 (* returns the smallest element in x::lst *)
 let min_in_sublist x lst =
    List.fold_left _________ (* #1 *)

 let smallest lst =
```

```
    List.map (fun sublist ->
      match sublist with
      | _____ (* #2 *) -> 0
      | _____ (* #3 *) -> min_in_sublist h t
    ) lst
```

#1

Enter your answer here

#2

Enter your answer here

#3

Enter your answer here

Save Answer

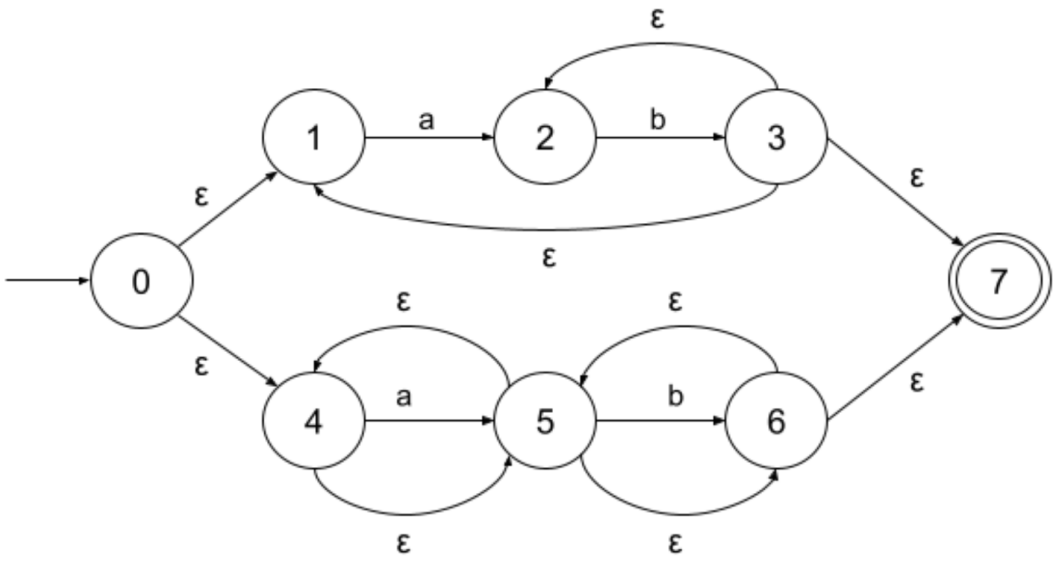
Q4 DFA/NFA

15 Points

Q4.1 NFA to Regex

3 Points

Give a regular expression equivalent to the following NFA.



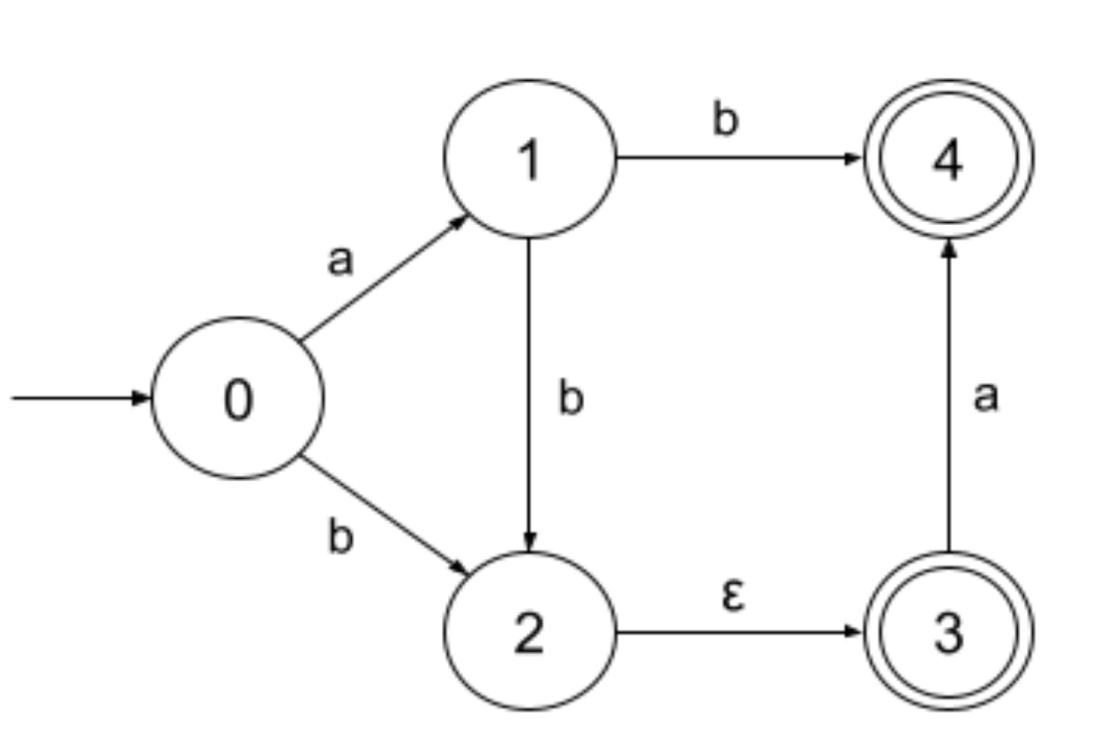
Enter your answer here

Save Answer

Q4.2 NFA accept

2 Points

Which of the following strings does this NFA accept? Check all that apply.



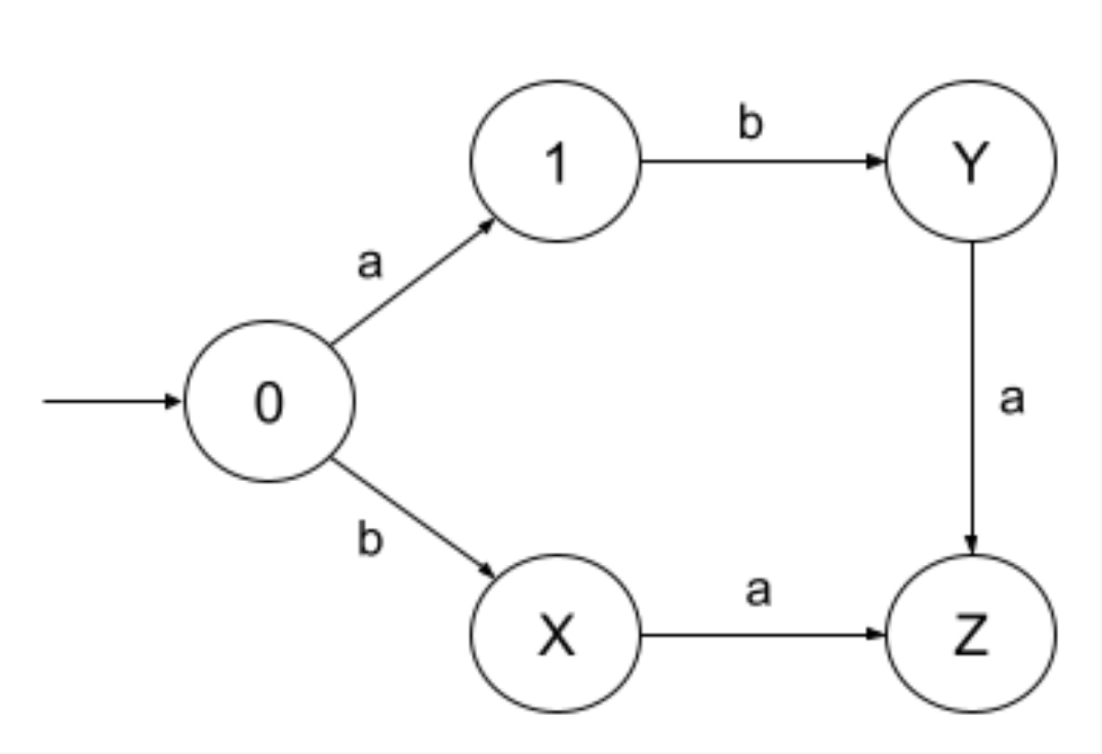
- [ ] ab
- [ ] bb
- [ ] aba
- [ ] bab

Save Answer

Q4.3 NFA to DFA

2 Points

The NFA from the question above can convert to the following equivalent DFA using the subset construction algorithm (final states have not been marked):



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

- [ ] 0
- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4

Save Answer

Q4.4 NFA to DFA

2 Points

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

- [ ] 0
- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4

Save Answer

Q4.5 NFA to DFA

2 Points

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

- [ ] 0
- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4

Save Answer

Q4.6 NFA to DFA

2 Points

In this DFA, which states are final states?

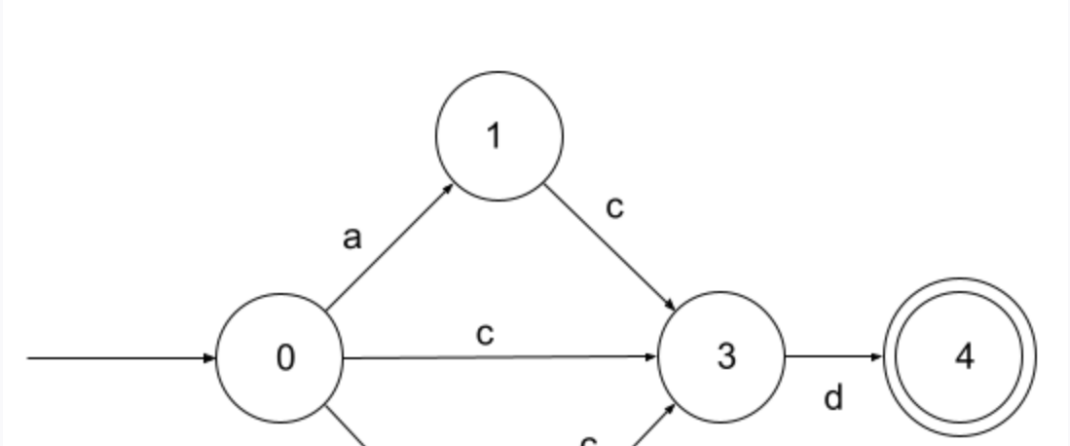
- [ ] 0
- [ ] 1
- [ ] X
- [ ] Y
- [ ] Z

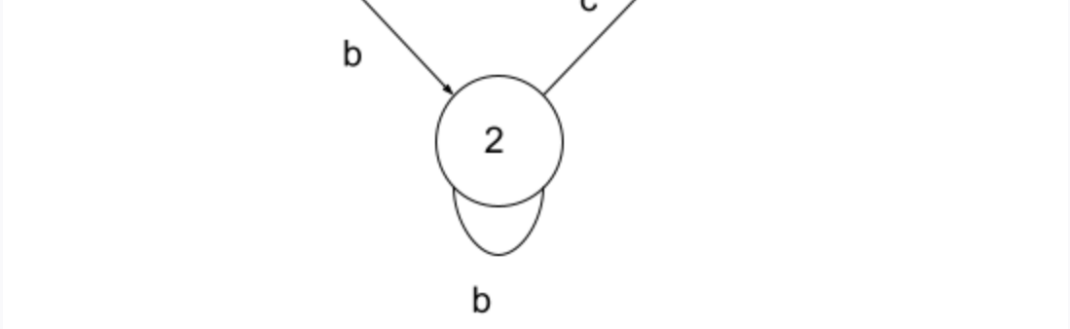
Save Answer

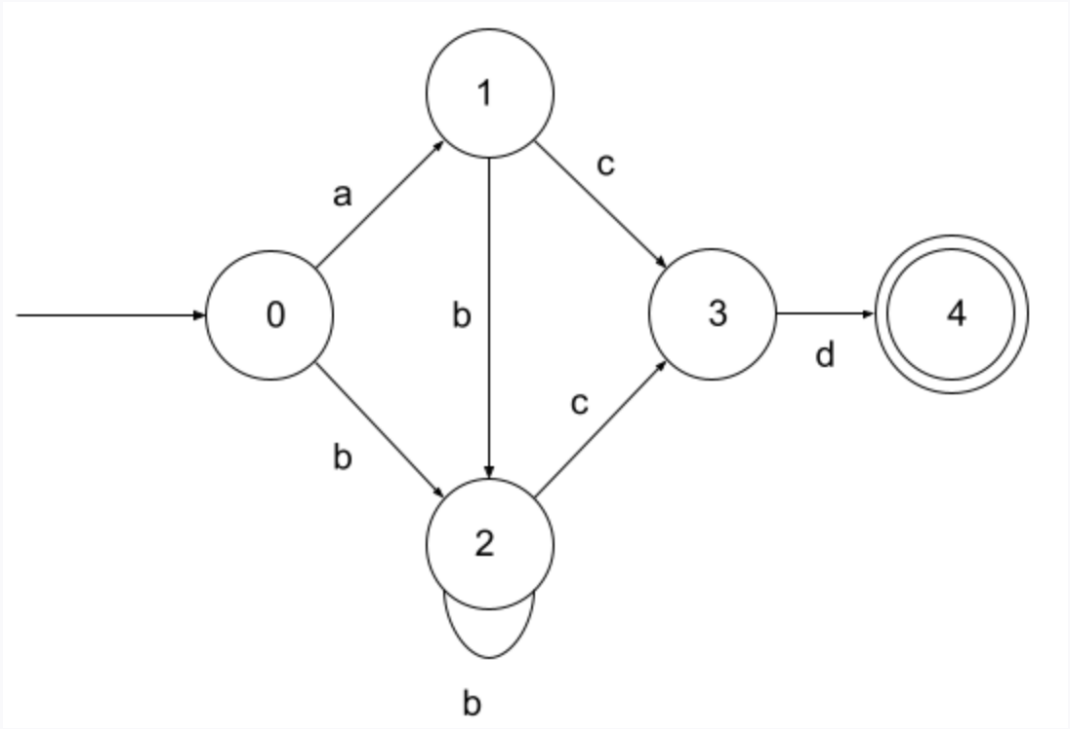
Q4.7 Regex/DFA

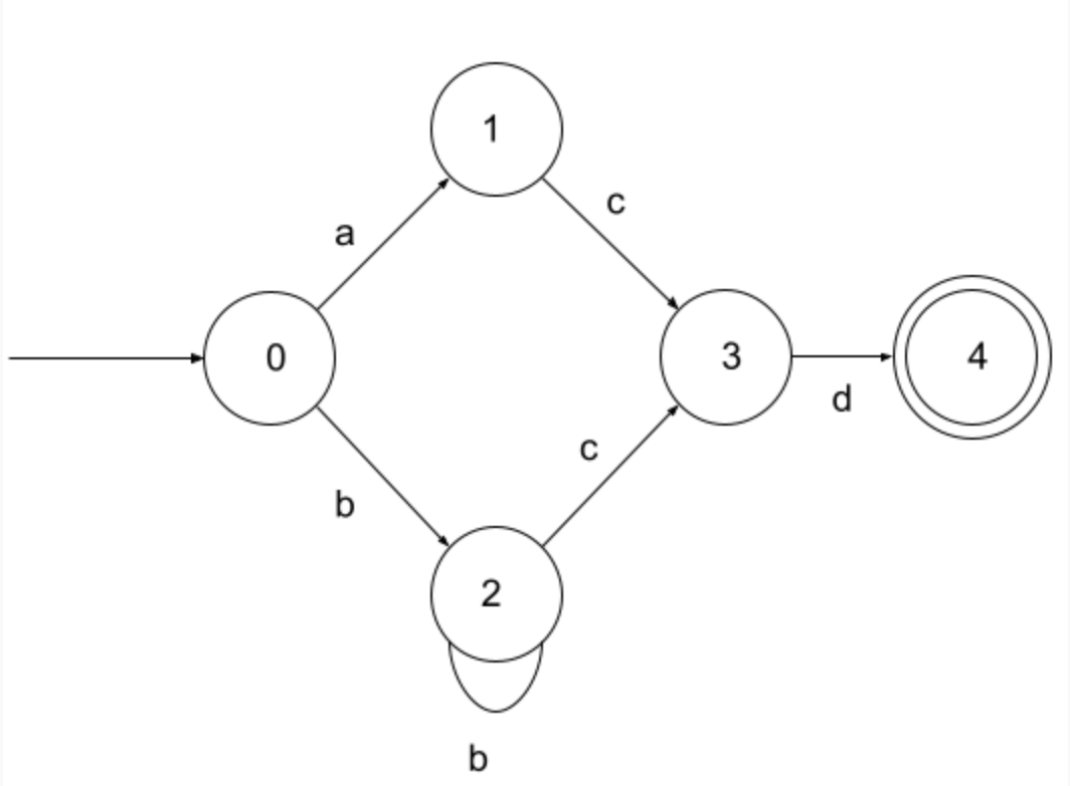
2 Points

Which of the following DFAs recognizes the language denoted by the regular expression `(a|b+)cd` ?

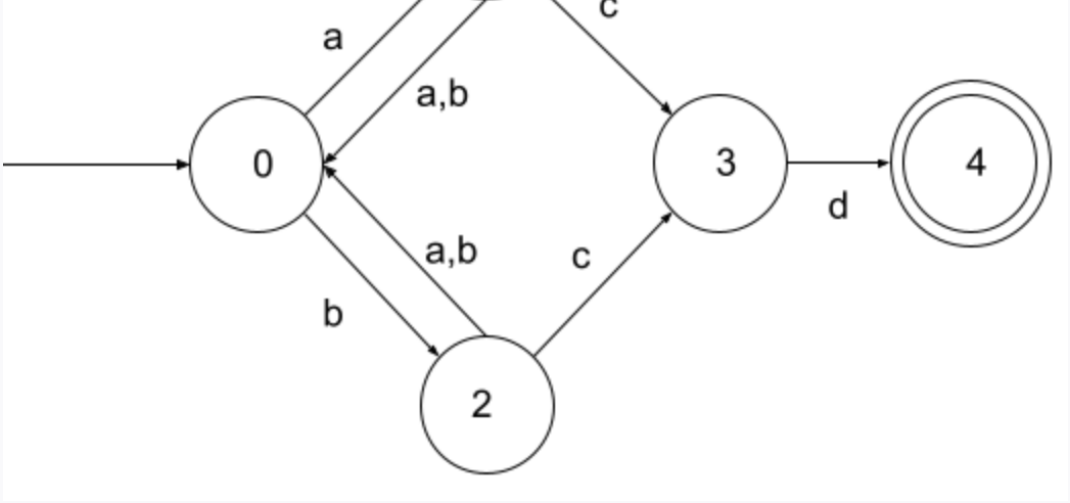



b
2
b


1
a
c
0
b
3
4
d
c
b
2
b


1
a
c
0
3
4
d
c
b
2
b


1



Save Answer

Q5 CFG and Parsing

9 Points

Q5.1 Language expressiveness

1 Point

If a language can be described by a CFG, then it can also be described by a DFA or NFA

- True
- False

Save Answer

Q5.2 Parse trees

1 Point

If a CFG accepts a string s, then there is always a unique parse tree for s.

- True
- False

Save Answer

Q5.3 Ambiguity

4 Points

Prove the following CFG is ambiguous by showing two distinct derivations of `ccc`. Nonterminals are in capital letters, and terminals are in lowercase. S is the start symbol.

```
S → SS | T | aS
T → bT | c | TT | S
```

Enter your answer here

Save Answer

Q5.4 Derivations

3 Points

Using the simplified version of the CFG from Project 4A given below, give the **leftmost derivation** of `x * 2`. In the CFG, nonterminals are in *italics* and terminals are in `typewriter` font. *Exp* is the start symbol.

Exp → *Add*
Add → *Mul* `+` *Add* | *Mul*
Mul → *Primary* `*` *Mul* | *Primary*
Primary → *Int* | *ID* | `(` *Exp* `)`
Int → `0` | `1` | `2`
ID → `x` | `y`

Enter your answer here

Save Answer

Q6 Security

7 Points

Q6.1 Fibonacci

5 Points

Consider the following code snippet (written in C) used to calculate Fibonacci numbers:

```
0   int fib(int n) {
1     int a[50];
2     int i;
3     a[0] = 1;
4     a[1] = 1;
5     for (i = 1; i < n; i++) {
6       a[i+1] = a[i]+a[i-1];
7     }
8     for (i = 0; i < n; i++) {
9       printf("%d,", a[i]);
10    }
11  }
```

This code has a bug in it that makes it potentially vulnerable to attack.

What kind of bug is it?

- ○ Use after free
- ○ Buffer overflow
- ○ SQL injection
- ○ Command injection

What input `n` could you provide to `fib` to cause the bug to be triggered?

Enter your answer here

Suppose you wanted to use *input sanitization* to prevent such a trigger from taking place. How would you do this (be specific, e.g., using line numbers) ?

Enter your answer here

Save Answer

Q6.2 Type Safety

2 Points

Type safe languages like Java and OCaml are not subject to buffer overflows. Explain Why?

Enter your answer here

Save Answer

Q7 Rust

18 Points

Q7.1 Finish the code

6 Points

Consider the following Rust code.

```
fn looper(n: ________) {    // #1
  let _______ i = n;        // #2
  while i >= 0 {
    println!("{}", i);
    _________               // #3
  }
}

fn main() {
  looper(3);
}
```

Fill in the three blanks so that when run the program prints

```
3
2
1
0
```

#1

Enter your answer here

#2

Enter your answer here

#3

Enter your answer here

Save Answer

Q7.2 Finish the code

4 Points

Fill in the blank in the following code.

```
pub enum Flavor {
    Sweet,
    Savory
}
pub struct Recipe<'a> {
  name: &'a str,
  flavor: Flavor,
}

pub trait Summarizable {
  fn summary(&self) -> String;
}
impl <'a> Summarizable for Recipe<'a> {
    fn summary(&self) -> String {
        let s = _____________;  // #1
        format!("{} is {}",self.name,s)
    }
}

pub fn main() {
    let r1 = Recipe { name: "jam", flavor: Flavor::Sweet };
    let r2 = Recipe { name: "marmite", flavor: Flavor::Savory };
    println!("{}, {}", r1.summary(), r2.summary());
}
```

When the `main` function is executed, it should print

```
jam is sweet, marmite is savory
```

#1

Enter your answer here

Save Answer

Q7.3 Ownership

1 Point

Consider the following Rust program:

```
fn main() {
  let a = String::from("A");
  {
    let b = String::from("B");
    let c = &a;
    // HERE 1
    {
      let d = b;
      println!("{}", d);
    }
    // HERE 2
    println!("{}", c);
  }
}
```

Which variable owns the string "A" at the point marked `HERE 1`?

- a
- b
- c
- none -- there's type error

Save Answer

Q7.4 Ownership

0 Points

Which variable owns the string "B" at the point marked `HERE 2`?

- b
- c
- d
- none -- there's a type error

Save Answer

Q7.5 Borrowing

3 Points

Consider the following Rust program:

```
fn main() {
  let my_string = String::from("hello");
  do_stuff(&my_string);
  assert_eq!("hello world", my_string);
}

fn do_stuff(s: &String) {
  s.push_str(" world");
}
```

The program fails to compile. Briefly explain what we have to change in order to make it work as expected (i.e., pass the assertion).

Enter your answer here

Save Answer

Q7.6 Reference counting

2 Points

Consider:

```
1  fn main() {
2    let x = String::from("cmsc330");
3    let r1 = Rc::new(x);
4    let r2 = Rc::clone(&r1);
5    let mut r3;
6    {
7      let r4 = Rc::clone(&r2);
8      r3 = Some(r2);
9    }
10   r3 = None;
11 }
```

On line 3, an `Rc` object is created. On which lines is its refcount increased?

- [ ] 4
- [ ] 5
- [ ] 7
- [ ] 8
- [ ] 10

Save Answer

Q7.7 Reference counting
2 Points

On which lines is the count decreased?

- [ ] 7
- [ ] 8
- [ ] 9
- [ ] 10
- [ ] 11

Save Answer

Q8 Lambda Calculus
9 Points

Q8.1 Alpha equivalence
2 Points

Select all of the options that are alpha equivalent to the given expression:

```
(λx. x (λy. x)) x
```

- [ ] `(λw . w (λy . w)) x`
- [ ] `(λz . z (λy . x)) x`
- [ ] `(λa . a (λy . a)) a`
- [ ] `(λx . x (λa . x)) x`

Save Answer

Q8.2 Evaluation strategy
2 Points

Consider this lambda term

```
(λx. x) ((λy. y) z)
```

Which of the following is a *lazy* evaluation of this term?

- `(λx. x) ((λy. y) z)` → `((λy. y) z)` → `z`
- `(λx. x) ((λy. y) z)` → `(λx. x) z` → `z`
- `(λx. x) ((λy. y) z)` → `(λx. x) y` → `y`
- `(λx. x) ((λy. y) z)` → `(λx. x) (λy. y)` → `x`

Save Answer

Q8.3 Evaluation strategy

2 Points

Consider the same lambda term as above:

```
(λx. x) ((λy. y) z)
```

Which of the following is an *eager* evaluation of this term?

- `(λx. x) ((λy. y) z)` → `((λy. y) z)` → `z`
- `(λx. x) ((λy. y) z)` → `(λx. x) z` → `z`
- `(λx. x) ((λy. y) z)` → `(λx. x) y` → `y`
- `(λx. x) ((λy. y) z)` → `(λx. x) (λy. y)` → `x`

Save Answer

Q8.4 Evaluation

2 Points

Consider the following lambda term:

```
(λx. x x) (λx. x x)
```

How many beta reductions will this term take before it reaches a normal form?

- 0
- 1
- 2
- it never reaches a normal form

Save Answer

Q8.5 Combinators

1 Point

The *Y combinator* is used for what purpose?

- To implement recursion
- To implement dynamic scoping
- To implement closures
- To avoid shadowing

Save Answer

Save All Answers

Submit & View Submission >