CMSC330 Spring 2025 Quiz 4 - SOLUTIONS

Proctoring TA: _____ **Name:** _____

Section Number: _____ **UID:** _____

Problem 1: Concepts

[Total 4 pts]

	true	false
There exist safe programs that Rust will not compile.	(T)	F
Performing Eager and Lazy evaluation on the same Lambda Calculus expression will always result in the same beta normal form (assuming the expression terminates).	(T)	F
In Rust, you can have multiple mutable references with overlapping lifetimes that point to the same piece of data.	T	(F)
Lambda calculus can mimic the behavior of a finite state machine.	(T)	F
There exist safe programs that Rust will not compile.	(T)	F
Lambda calculus can mimic the behavior of a finite state machine.	(T)	F
Performing Eager and Lazy evaluation on the same Lambda Calculus expression will always result in the same beta normal form (assuming the expression terminates).	(T)	F
In Rust, you can have multiple mutable references with overlapping lifetimes that point to the same piece of data.	T	(F)
Lambda calculus can mimic the behavior of a finite state machine.	(T)	F
In Rust, you can have multiple mutable references with overlapping lifetimes that point to the same piece of data.	T	(F)
There exist safe programs that Rust will not compile.	(T)	F
Performing Eager and Lazy evaluation on the same Lambda Calculus expression will always result in the same beta normal form (assuming the expression terminates).	(T)	F
In Rust, you can have multiple mutable references with overlapping lifetimes that point to the same piece of data.	T	(F)
There exist safe programs that Rust will not compile.	(T)	F
Lambda calculus can mimic the behavior of a finite state machine.	(T)	F
Performing Eager and Lazy evaluation on the same Lambda Calculus expression will always result in the same beta normal form (assuming the expression terminates).	(T)	F

Problem 2: Lambda Calculus

[Total 8 pts]

(a) Reduce [3 pts]

Reduce the following lambda expression to beta normal form using **eager evaluation**. Show every step, including alpha conversions, if you used any.

$$(\lambda a.\ (\lambda y.\ y\ b)\ a)\ ((\lambda y.\ y)\ a)$$

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

(b) Free Variables: [3 pts]

Circle the free variables in the expression below:

$$((\lambda b.\ (\lambda y.\ \text{(a)}\ y)\ (\lambda z.\ \text{(y)}\ b))\ \text{(b)})\ ((\lambda r.\ r\ \text{(a)})\ \text{(b)})$$

(c) Alpha Equivalence: [2 pts]

Which of the following are alpha equivalent to the following expression (from part b): $((\lambda b.\ (\lambda y.\ a\ y)\ (\lambda z.\ y\ b))\ b)\ ((\lambda r.\ r\ a)\ b)$? **Select all that apply.**

(A) $((\lambda m.\ (\lambda n.\ r\ n)\ (\lambda p.\ n\ p))\ z)\ ((\lambda q.\ q\ r)\ z)$
(B) $((\lambda m.\ (\lambda n.\ a\ n)\ (\lambda p.\ y\ m))\ b)\ ((\lambda q.\ q\ a)\ b)$
(C) $((\lambda m.\ (\lambda n.\ a\ n)\ (\lambda p.\ y\ m))\ m)\ ((\lambda q.\ q\ a)\ m)$
(D) $((\lambda m.\ (\lambda n.\ a\ n)\ (\lambda p.\ n\ m))\ b)\ ((\lambda q.\ q\ a)\ b)$

Problem 3: Lambda Calculus

[Total 8 pts]

(a) Reduce [3 pts]

Reduce the following lambda expression to beta normal form using **eager evaluation**. Show every step, including alpha conversions, if you used any.

$$(\lambda c.\ (\lambda y.\ y\ d)\ c)\ ((\lambda y.\ y)\ c)$$

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

(b) Free Variables: [3 pts]

Circle the free variables in the expression below:

$$((\lambda d.\ (\lambda y.\ \text{c}\ y)\ (\lambda z.\ \text{y}\ d))\ \text{d})\ ((\lambda r.\ r\ \text{c})\ \text{d})$$

(c) Alpha Equivalence: [2 pts]

Which of the following are alpha equivalent to the following expression (from part b): $((\lambda d.\ (\lambda y.\ c\ y)\ (\lambda z.\ y\ d))\ d)\ ((\lambda r.\ r\ c)\ d)$? **Select all that apply.**

(A) $((\lambda m.\ (\lambda n.\ r\ n)\ (\lambda p.\ n\ p))\ z)\ ((\lambda q.\ q\ r)\ z)$
(B) $((\lambda m.\ (\lambda n.\ c\ n)\ (\lambda p.\ y\ m))\ d)\ ((\lambda q.\ q\ c)\ d)$
(C) $((\lambda m.\ (\lambda n.\ c\ n)\ (\lambda p.\ y\ m))\ m)\ ((\lambda q.\ q\ c)\ m)$
(D) $((\lambda m.\ (\lambda n.\ c\ n)\ (\lambda p.\ n\ m))\ d)\ ((\lambda q.\ q\ c)\ d)$

Problem 4: Lambda Calculus [Total 8 pts]

(a) Reduce [3 pts]

Reduce the following lambda expression to beta normal form using **eager evaluation**. Show every step, including alpha conversions, if you used any.

$$(\lambda e.\ (\lambda y.\ y\ f)\ e)\ ((\lambda y.\ y)\ e)$$

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

(b) Free Variables: [3 pts]
Circle the free variables in the expression below:

$$((\lambda f.\ (\lambda y.\ \text{e}\ y)\ (\lambda z.\ \text{y}\ f))\ \text{f})\ ((\lambda r.\ r\ \text{e})\ \text{f})$$

(c) Alpha Equivalence: [2 pts]

Which of the following are alpha equivalent to the following expression (from part b): $((\lambda f.\ (\lambda y.\ e\ y)\ (\lambda z.\ y\ f))\ f)\ ((\lambda r.\ r\ e)\ f)$? **Select all that apply.**

- (A) $((\lambda m.\ (\lambda n.\ r\ n)\ (\lambda p.\ n\ p))\ z)\ ((\lambda q.\ q\ r)\ z)$
- (B) $((\lambda m.\ (\lambda n.\ e\ n)\ (\lambda p.\ y\ m))\ f)\ ((\lambda q.\ q\ e)\ f)$
- (C) $((\lambda m.\ (\lambda n.\ e\ n)\ (\lambda p.\ y\ m))\ m)\ ((\lambda q.\ q\ e)\ m)$
- (D) $((\lambda m.\ (\lambda n.\ e\ n)\ (\lambda p.\ n\ m))\ f)\ ((\lambda q.\ q\ e)\ f)$

Problem 5: Lambda Calculus [Total 8 pts]

(a) Reduce [3 pts]
Reduce the following lambda expression to beta normal form using **eager evaluation**. Show every step, including alpha conversions, if you used any.

$$(\lambda j.\ (\lambda y.\ y\ k)\ j)\ ((\lambda y.\ y)\ j)$$

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

(b) Free Variables: [3 pts]
Circle the free variables in the expression below:

$$((\lambda k.\ (\lambda y.\ \text{j}\ y)\ (\lambda z.\ \text{y}\ k))\ \text{k})\ ((\lambda r.\ r\ \text{j})\ \text{k})$$

(c) Alpha Equivalence: [2 pts]

Which of the following are alpha equivalent to the following expression (from part b): $((\lambda k.\ (\lambda y.\ j\ y)\ (\lambda z.\ y\ k))\ k)\ ((\lambda r.\ r\ j)\ k)$? **Select all that apply.**

- (A) $((\lambda m.\ (\lambda n.\ r\ n)\ (\lambda p.\ n\ p))\ z)\ ((\lambda q.\ q\ r)\ z)$
- (B) $((\lambda m.\ (\lambda n.\ j\ n)\ (\lambda p.\ y\ m))\ k)\ ((\lambda q.\ q\ j)\ k)$
- (C) $((\lambda m.\ (\lambda n.\ j\ n)\ (\lambda p.\ y\ m))\ m)\ ((\lambda q.\ q\ j)\ m)$
- (D) $((\lambda m.\ (\lambda n.\ j\ n)\ (\lambda p.\ n\ m))\ k)\ ((\lambda q.\ q\ j)\ k)$

Problem 6: Rust Ownership

[Total 8 pts]

```
fn main(){
  {
    let mut a = String::from("330");
    // Mark 1

    let mut b = f1(&mut a);
    // Mark 2

    let d = String::from("330slays");
    let c = f2(d);
    // Mark 3

    println!("{c}");
  }
  // Mark 4
}

fn f1(s: &mut String)-> () {
    s.push_str("slays");
    // Mark 5
}

fn f2(s: String) -> String {
  println!("{}",s.len());
  // Mark 6
  s
}
```

If there is no owner (because the value has been dropped) put "None". Assume that we are asking about ownership **during** execution.

Who is the owner of the value "330" at Mark 1?

a

Who is the owner of the value "330slays" at Mark 2?

a

Who is the owner of the value "330slays" at Mark 3?

c/ a

Who is the owner of the value "330slays" at Mark 4?

None

Who is the owner of the value "330slays" at Mark 5?

a

Who is the owner of the value "330slays" at Mark 6?

s

What is printed out when this program is run?

8
330*slays*

Problem 7: Rust Ownership

[Total 8 pts]

```
fn main(){
  {
    let mut x = String::from("meow");
    // Mark 1

    let mut y = f1(&mut x);
    // Mark 2

    let w = String::from("meowbark");
    let z = f2(w);
    // Mark 3

    println!("{z}");
  }
  // Mark 4
}

fn f1(s: &mut String)-> () {
    s.push_str("bark");
    // Mark 5
}

fn f2(s: String) -> String {
  println!("{}",s.len());
  // Mark 6
  s
}
```

If there is no owner (because the value has been dropped) put "None". Assume that we are asking about ownership **during** execution.

Who is the owner of the value "meow" at Mark 1?

x

Who is the owner of the value "meowbark" at Mark 2?

x

Who is the owner of the value "meowbark" at Mark 3?

z/ x

Who is the owner of the value "meowbark" at Mark 4?

None

Who is the owner of the value "meowbark" at Mark 5?

x

Who is the owner of the value "meowbark" at Mark 6?

s

What is printed out when this program is run?

8
meowbark

Problem 8: Rust Ownership

[Total 8 pts]

```
fn main(){
  {
    let mut a = String::from("330");
    // Mark 1

    let mut b = f1(&mut a);
    // Mark 2

    let d = String::from("330slays");
    let c = f2(d);
    // Mark 3

    println!("{c}");
  }
  // Mark 4
}

fn f1(s: &mut String)-> () {
    s.push_str("slays");
    // Mark 5
}

fn f2(s: String) -> String {
  println!("{}",s.len());
  // Mark 6
  s
}
```

If there is no owner (because the value has been dropped) put "None". Assume that we are asking about ownership **during** execution.

Who is the owner of the value "330" at Mark 1?

a

Who is the owner of the value "330slays" at Mark 2?

a

Who is the owner of the value "330slays" at Mark 3?

c/ a

Who is the owner of the value "330slays" at Mark 4?

None

Who is the owner of the value "330slays" at Mark 5?

a

Who is the owner of the value "330slays" at Mark 6?

s

What is printed out when this program is run?

8
330*slays*

Problem 9: Rust Ownership

[Total 8 pts]

```
fn main(){
  {
    let mut x = String::from("meow");
    // Mark 1

    let mut y = f1(&mut x);
    // Mark 2

    let w = String::from("meowbark");
    let z = f2(w);
    // Mark 3

    println!("{z}");
  }
  // Mark 4
}

fn f1(s: &mut String)-> () {
    s.push_str("bark");
    // Mark 5
}

fn f2(s: String) -> String {
  println!("{}",s.len());
  // Mark 6
  s
}
```

If there is no owner (because the value has been dropped) put "None". Assume that we are asking about ownership **during** execution.

Who is the owner of the value "meow" at Mark 1?

x

Who is the owner of the value "meowbark" at Mark 2?

x

Who is the owner of the value "meowbark" at Mark 3?

z/ x

Who is the owner of the value "meowbark" at Mark 4?

None

Who is the owner of the value "meowbark" at Mark 5?

x

Who is the owner of the value "meowbark" at Mark 6?

s

What is printed out when this program is run?

8
meowbark