



Proctoring TA:	Name:
Section Number:	UID:

Problem 1: Concepts [Total 4 pts] true false There exist safe programs that Rust will not compile. Performing Eager and Lazy evaluation on the same Lambda Calculus expression will always result in the same beta normal form (assuming the expression terminates). In Rust, you can have multiple mutable references with overlapping lifetimes that point to the same piece of data. Lambda calculus can mimic the behavior of a finite state machine. There exist safe programs that Rust will not compile. Lambda calculus can mimic the behavior of a finite state machine. Performing Eager and Lazy evaluation on the same Lambda Calculus expression will always result in the same beta normal form (assuming the expression terminates). In Rust, you can have multiple mutable references with overlapping lifetimes that point to the same piece of data. Lambda calculus can mimic the behavior of a finite state machine. In Rust, you can have multiple mutable references with overlapping lifetimes that point to the same piece of data. There exist safe programs that Rust will not compile. Performing Eager and Lazy evaluation on the same Lambda Calculus expression will always result in the same beta normal form (assuming the expression terminates). In Rust, you can have multiple mutable references with overlapping lifetimes that point to the same piece of data. There exist safe programs that Rust will not compile. Lambda calculus can mimic the behavior of a finite state machine. Performing Eager and Lazy evaluation on the same Lambda Calculus expression will always result in the same beta normal form (assuming the expression terminates).

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:

$$((\lambda b. (\lambda y. a) y) (\lambda z. y) b) b) ((\lambda r. ra) b)$$

(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) $((\lambda m. (\lambda n. a n) (\lambda p. y m)) m) ((\lambda q. q a) m)$
- $(D) ((\lambda m. (\lambda n. a n) (\lambda p. n m)) b) ((\lambda q. q a) b)$

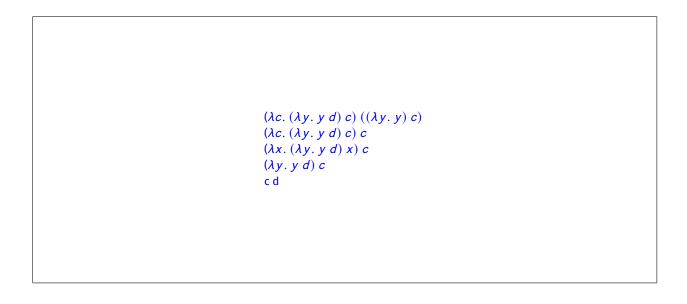
Problem 3: Lambda Calculus

[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.

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



(b) Free Variables: [3 pts]

Circle the free variables in the expression below:

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

(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.

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

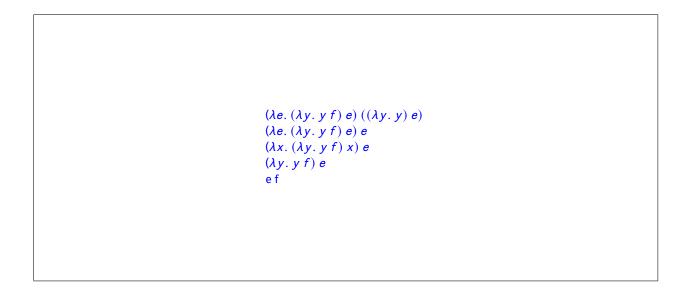
Problem 4: Lambda Calculus

[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.

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



(b) Free Variables: [3 pts]

Circle the free variables in the expression below:

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

(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) $((\lambda m. (\lambda n. e n) (\lambda p. y m)) m) ((\lambda q. q e) m)$

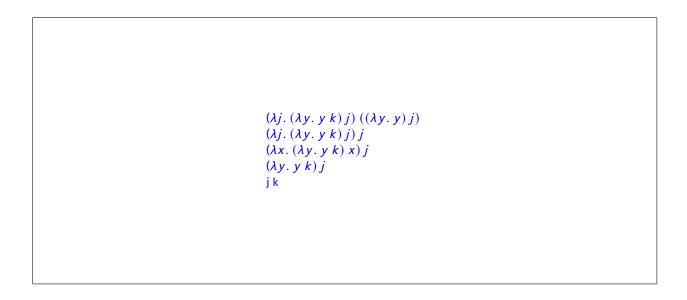
Problem 5: Lambda Calculus

[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.

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



(b) Free Variables: [3 pts]

Circle the free variables in the expression below:

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

(c) Alpha Equivalence: [2 pts]

Which of the following are alpha equivalent to the following expression (from part b): $((\lambda k. (\lambda y. j \ y) (\lambda z. y \ k)) \ k) ((\lambda r. rj) \ k)$? **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) $((\lambda m. (\lambda n. j n) (\lambda p. y m)) m) ((\lambda q. q j) m)$
- $(D) ((\lambda m. (\lambda n. j n) (\lambda p. n m)) k) ((\lambda q. q j) k)$

```
"None". Assume that we are asking about ownership during
fn main(){
                                             execution.
    let mut a = String::from("330");
                                            Who is the owner of the value "330" at Mark 1?
    // Mark 1
    let mut b = f1(&mut a);
    // Mark 2
                                            Who is the owner of the value "330slays" at Mark 2?
    let d = String::from("330slays");
    let c = f2(d);
    // Mark 3
                                             Who is the owner of the value "330slays" at Mark 3?
    println!("{c}");
                                               c/a
  // Mark 4
}
                                            Who is the owner of the value "330slays" at Mark 4?
                                               None
fn f1(s: &mut String)-> () {
    s.push_str("slays");
    // Mark 5
                                            Who is the owner of the value "330slays" at Mark 5?
}
fn f2(s: String) -> String {
  println!("{}",s.len());
                                            Who is the owner of the value "330slays" at Mark 6?
  // Mark 6
}
```

What is printed out when this program is run?

```
8
330slays
```

Problem 7: Rust Ownership

[Total 8 pts]

```
fn main(){
                                                 "None". Assume that we are asking about ownership during
                                                 execution.
         let mut x = String::from("meow");
                                                 Who is the owner of the value "meow" at Mark 1?
         // Mark 1
         let mut y = f1(\&mut x);
         // Mark 2
                                                 Who is the owner of the value "meowbark" at Mark 2?
         let w = String::from("meowbark");
         let z = f2(w);
         // Mark 3
                                                 Who is the owner of the value "meowbark" at Mark 3?
         println!("{z}");
                                                    z/x
       // Mark 4
    }
                                                 Who is the owner of the value "meowbark" at Mark 4?
                                                   None
    fn f1(s: &mut String)-> () {
         s.push_str("bark");
         // Mark 5
                                                 Who is the owner of the value "meowbark" at Mark 5?
    fn f2(s: String) -> String {
       println!("{}",s.len());
                                                 Who is the owner of the value "meowbark" at Mark 6?
       // Mark 6
                                                     S
    }
What is printed out when this program is run?
                                              meowbark
```

Problem 8: Rust Ownership

[Total 8 pts]

```
"None". Assume that we are asking about ownership during
    fn main(){
                                                  execution.
         let mut a = String::from("330");
                                                 Who is the owner of the value "330" at Mark 1?
         // Mark 1
         let mut b = f1(&mut a);
         // Mark 2
                                                 Who is the owner of the value "330slays" at Mark 2?
         let d = String::from("330slays");
         let c = f2(d);
         // Mark 3
                                                  Who is the owner of the value "330slays" at Mark 3?
         println!("{c}");
                                                     c/a
      }
       // Mark 4
    }
                                                 Who is the owner of the value "330slays" at Mark 4?
                                                    None
    fn f1(s: &mut String)-> () {
         s.push_str("slays");
         // Mark 5
                                                 Who is the owner of the value "330slays" at Mark 5?
    }
                                                      a
    fn f2(s: String) -> String {
      println!("{}",s.len());
                                                 Who is the owner of the value "330slays" at Mark 6?
      // Mark 6
      s
    }
What is printed out when this program is run?
                                                330slays
```

Problem 9: Rust Ownership

[Total 8 pts]

```
"None". Assume that we are asking about ownership during
fn main(){
                                            execution.
    let mut x = String::from("meow");
                                            Who is the owner of the value "meow" at Mark 1?
    // Mark 1
    let mut y = f1(\&mut x);
    // Mark 2
                                            Who is the owner of the value "meowbark" at Mark 2?
    let w = String::from("meowbark");
    let z = f2(w);
    // Mark 3
                                            Who is the owner of the value "meowbark" at Mark 3?
    println!("{z}");
                                               z/x
  // Mark 4
}
                                            Who is the owner of the value "meowbark" at Mark 4?
                                              None
fn f1(s: &mut String)-> () {
    s.push_str("bark");
    // Mark 5
                                            Who is the owner of the value "meowbark" at Mark 5?
fn f2(s: String) -> String {
  println!("{}",s.len());
                                            Who is the owner of the value "meowbark" at Mark 6?
  // Mark 6
                                                S
}
```

What is printed out when this program is run?

8 meowbark