CMSC330 Spring 2024 Quiz 4 Solutions

Problem 1: Basics

[Total 4 pts]

	True	False
There are some data structures in Rust which will not deallocate using the Reference Counting Garbage Collection Strategy	● T	○ F
There are some data structures in Rust which will deallocate using the Reference Counting Garbage Collection Strategy	● T	○ F
Rust's Type System prevents Double Frees unless the **unsafe** keyword is used	● T	○ F
Rust's Type System prevents Double Frees unless the **safe** keyword is used	○ T	● F
It is theoretically possible to implement project 3 (NFA to DFA) in Lambda Calculus	● T	○ F
It is theoretically possible to implement project 4 (MicroCaml) in Lambda Calculus	● T	○ F
It is impossible to implement project 4 (MicroCaml) in Lambda Calculus	○ T	● F
It is impossible to implement project 3 (NFA to DFA) in Lambda Calculus	○ T	● F
$(\lambda x.y)((\lambda x.xx)(\lambda z.zz))$ has a beta normal form under eager evaluation	○ T	● F
$(\lambda x.x)((\lambda y.yy)(\lambda z.zz))$ has a beta normal form under eager evaluation	○ T	● F
$(\lambda y.y)((\lambda y.xy)(\lambda z.zz))$ has a beta normal form under eager evaluation	● T	○ F
$(\lambda y.x)((\lambda x.xx)(\lambda y.xy))$ has a beta normal form under eager evaluation	● T	○ F

Problem 2: Lambda Calculus - Variables

[Total 2 pts]

Underline the free variables and circle the bound variables in the expression below.

Note: Do not mark any of the lambda parameter variables.

Version A:

$$\underline{a}\ (\lambda a.\ \lambda b.\ b\ \lambda a.\ a)(\lambda c.\ \underline{d})\ \underline{c}$$

Version B:

$$\underline{z}\ (\lambda f.\ (\lambda b.\ \underline{a}\ \lambda a.\ a))(\lambda c.\ c)\ \underline{c}$$

Version C:

$$(\lambda a.\ \lambda b.\ b)\ (\lambda a.\ a\ \underline{z}\ \lambda c.\ \underline{d})\ \underline{d}$$

Version D:

$$(\lambda x.\ \lambda b.\ x)\ (\lambda y.\ \underline{a}\ y\ \lambda c.\ c)\ \underline{d}$$

Problem 3: Lambda Calculus - Alpha Equivalence

[Total 2 pts]

Version A:

Which lambda calculus expressions are alpha equivalent to $(\lambda a.\ a)((\lambda b.\ c\ \lambda x.\ x)\ a\ b\ c)$? Circle all that apply.

A $(\lambda a.\ a)((\lambda a.\ c\ \lambda a.\ a)\ a\ b\ c)$ B $(c\ \lambda a.\ a)\ c$

C $(\lambda c.\ a)((\lambda b.\ c\ \lambda c.\ c)\ a\ b\ c)$ D $(\lambda f.\ f)((\lambda c.\ c\ \lambda g.\ g)\ a\ b\ c)$

Version B:

Which lambda calculus expressions are alpha equivalent to $(\lambda b.\ a)((\lambda c.\ c\ \lambda b.\ b)\ x\ y\ z)$? Circle all that apply.

A $(\lambda a.\ a)((\lambda b.\ c\ \lambda a.\ a)\ a\ b\ c)$ B $(\lambda x.\ a)((\lambda g.\ g\ \lambda a.\ a)\ x\ y\ z)$

C $(\lambda d.\ a)((\lambda c.\ c\ \lambda b.\ b)\ x\ y\ z)$ D $(\lambda b.\ b)((c\ \lambda a.\ a)\ b\ c)$

Version C:

Which lambda calculus expressions are alpha equivalent to $(\lambda a.\ b)((\lambda b.\ c\ \lambda c.\ c)\ a\ b\ c)$? Circle all that apply.

A $(\lambda x.\ b)((\lambda c.\ c\ \lambda d.\ d)\ a\ b\ c)$ B $(\lambda x.\ a)((\lambda g.\ g\ \lambda a.\ a)\ x\ y\ z)$

C $(\lambda a.\ b)((\lambda c.\ c\ \lambda c.\ c)\ a\ b\ c)$ D $(\lambda a.\ b)((\lambda x.\ c\ \lambda c.\ c)\ a\ b\ c)$

Version D:

Which lambda calculus expressions are alpha equivalent to $(\lambda b.\ b)((\lambda y.\ z\ \lambda d.\ d)\ x\ y\ z)$? Circle all that apply.

A $(\lambda c.\ c)((\lambda d.\ z\ \lambda d.\ d)\ x\ y\ z)$ B $(\lambda b.\ b)((\lambda f.\ z\ \lambda f.\ f)\ x\ y\ z)$

C $(\lambda a.\ b)((\lambda c.\ c\ \lambda c.\ c)\ a\ b\ c)$ D $(\lambda a.\ b)((\lambda x.\ c\ \lambda c.\ c)\ a\ b\ c)$

Problem 4: Lambda Calculus - Reduction

[Total 4 pts]

Reduce the given lambda expression to beta normal form and show each step.

Version A:

Reduce $(\lambda a.(\lambda b.(\lambda c.c\ c)b)a)d$

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

Version B:

Reduce $(\lambda a.(\lambda b.(\lambda c.c\ f)b)a)x$

$(\lambda b.(\lambda c.c\ f)b)x$
$(\lambda c.c\ f)x$
$(x\ f)$

Version C:

Reduce $(\lambda x.(\lambda y.(\lambda z.z\ z)y)x)a$

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

Version D:

Reduce $(\lambda x.(\lambda y.(\lambda z.z\ f)y)x)d$

$(\lambda y.(\lambda z.z\ f)y)d$
$(\lambda z.z\ f)d$
$(d\ f)$

Problem 5: Rust Ownership

[Total 8 pts]

Version A, C:

```
fn main(){
  {
    let a = String::from("hello");
    let b = f1(a);
    // Mark 1
    let c = f2(&b);
    // Mark 2
  }
  // Mark 3
}

fn f1(s: String) -> String{
  println!("{}",s.len());
  // Mark 4
  s
}

fn f2(s: &str)-> i32{
    s.len() as i32
}
```

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 "hello" at Mark 1?

b

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

b

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

None

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

s

Version B, D:

```
fn main(){
  {
    let a = String::from("hello");
    // Mark 1
    let b = f1(a);
    // Mark 2
    let c = f2(&b);
    // Mark 3
  }
}

fn f1(s: String) -> String{
  println!("{}",s.len());
  s
  // Mark 4
}

fn f2(s: &str)-> i32{
    s.len() as i32
}
```

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 "hello" at Mark 1?

a

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

b

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

b

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

s/b