Chris Fenton
Fenwil28
CNC - Proglang
Midterm

**1. For the following program in C, is this type correct? What is its meaning? Identify any problems and how you might address them as a language designer. Give supporting reasons.**

```
1 int main ( ) {
2      int j = 0 ;
3      int i = 3 / j ;
4      for ( i =1; i > -1; i++)
5             i --;
6 }
```

I think this might be type correct in C, as I believe you can assign values during variable declaration; however, the value that is assigned to i in line 3 is undefined. If C uses some kind of NAN representation for divide by zero and other undefined values for ints, then maybe it passes the type checker. If this is the case, the meaning may still be murky if it results in a run-time error. Or maybe it doesn't matter because the assignment statement inside the for loop on line 4 overwrites the value for i (instead of declaring a scoped variable) to 1. In this the case, the loop still never stops.

**2. For the following program in Java, is this type correct? What is its meaning? Identify any problems and how you might address them as a language designer. Give supporting reasons.**

```
1 public class Sample {
2      public static void main (String args[]) {
3             int j;
4             int i;
5             j = 0;
6             if (j == 0)
7                    i = 5;
8             while (i > 0) {
9                    j = j + 1;
10            }
11     }
12 }
```

This looks type correct to me. Since the main method is type void, it doesn't return a value, so the meaning is just the state. In this case, it would be another infinite loop with states: [<j,0>,<i,5>], [<j,1>,<i,5>],[<j,2>,<i,5>]...

**3. Modify type rules 6.5 and 6.6 so that they check the validity and compute the type of an ArrayRef.**

**Rule 6.5**
An ArrayRef is valid if all of the following are true:

(a) If it's id appears in the typemap
(b) If it's index <= size

**Rule 6.6**
If the Expression is an ArrayRef, then its result type is the type of that ArraryRef.

**4. Show how the meaning of each of the following expressions and given states are derived from the semantic rules given in Section 8.2.3 (show the sequence of calls to the meaning function M and the current state pairs).**

**(a) M((x + 2) * y, {<x,2>, <y,-3>, <z,75>})**

M : Expression x State s -> value, so the meaning of this expression should be a value
(2 + 2) * -3 = -12
M((x + 2) * y, {<x,2>, <y,-3>, <z,75>}) = -12

**(b) M(2 * x + 3/y - 4, {<x,2>, <y,-3>, <z,75>})**
M : Expression x State s -> value, so the meaning of this expression should be a value
2 * 2 + 3/(-3) - 4 = 4 + (-1) - 4 = -1
M(2 * x + 3/y - 4, {<x,2>, <y,-3>, <z,75>}) = -1

**(c) M(1, {<x,2>, <y,-3>, <z,75>})**
According to meaning rule A.7 1: if the Expression is a Value, then its meaning is the meaning of the Value itself, so:
M(1, {<x,2>, <y,-3>, <z,75>}) = 1

**5. Modify the definition of the operators && and || so that they do not use short circuit evaluation. Give the new semantic rules.**

If the operator is a Boolean operator, then:

(a) The operator && is interpreted as:
a && b == *if* a *and* b *then* true *else* false
(b) The operator || is interpreted as:
a || b == *if* a *then* true
*if* b *then* true
*else* false
*else* false

**6. Assume we have that natural numbers as given in the lectures:**

**Z = λ f .λ x. x**
**S =** λ n. λ f. λx. f (n f x)

(the definitions above are for ease of use, so we can use the meta-characters Z and S even though there are no definitions in our variant of the lambda calculus.)

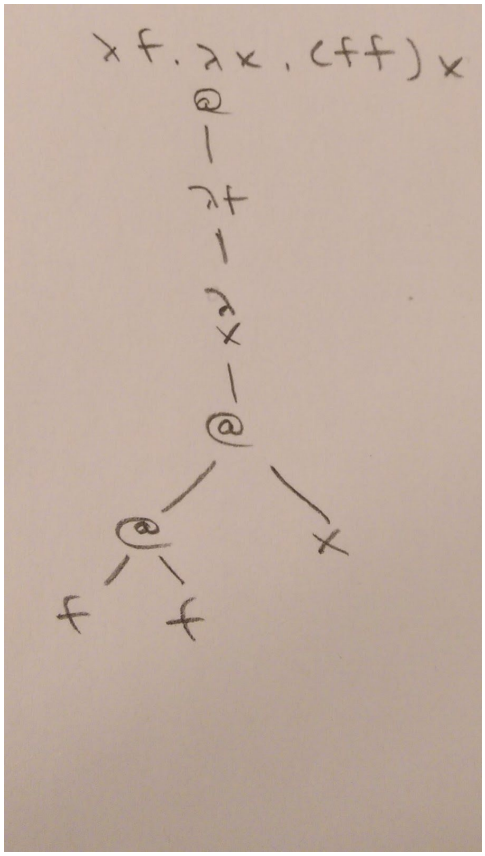
(a) Now show how to create a lambda term for One, using S and Z and showing the Beta-reductions. Be sure to identify the redexes (which expressions you are reducing at each step, using leftmost-outermost reduction order). Underline the redex at each step.

S Z → (λn.λf.λx) Z
→ (λn.λf.λx) (λ n. λ f. λx. f (n f x))
→ λf.λx. f x

(b) Is the following expression correctly formed? and if so give the parse tree:

λ f . λ x . ( f f ) x

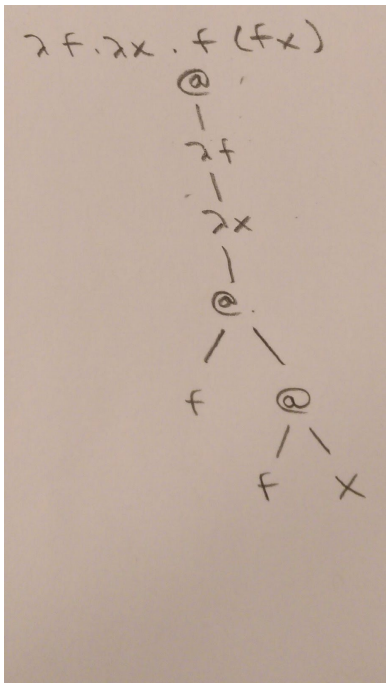
Yes



(c) Is the following  expression correctly formed? and if so give the parse tree):

λ f . λ x . f ( f x )

Yes.




(d) How would you restrict the language so that only one of these terms is correct?

**7. For the following lambda term:**

(λ x. y) ((λ x. x x) (λ x. x x))

(a) Show the 2 redexes (underline each separately).

(λ x. y) ((λ <u>x</u>. x x) (λ <u>x</u>. x x))

(b) Show the normal form using normal-order (leftmost-outermost) reduction.

(λ x. y) ((λ x. x x) (λ x. x x)) → (λ x. y) (x) (λ x. x x)
→ (λ x. y) (x) (x)
→ x y

(c) Show there is no normal form using applicative-order (leftmost-innermost).

(λ x. y) ((λ x. x x) (λ x. x x)) → (λ x. y) (λ x. x x) (x)
→ (λ x. y) (x) (x)
→ x y

**8.**
(a) Here is a definition of pred:

pred = λ n. λ f. λ x. n (λ g. λh. h (g f)) (λ u. x) (λ u. u)

Demonstrate how pred works on Z and one.

pred Z = λ n. λ f. λ x. n (λ g. λh. h (g f)) (λ u. x) (λ u. u) (λ f. λ x. x)
Not exactly sure how to do this, but the function doesn't get applied resulting in zero

one = λ f. λ x. f x
pred one = λ n. λ f. λ x. n (λ g. λh. h (g f)) (λ u. x) (λ u. u) (λ f. λ x. f x)
I get lost doing beta reductions at this scale :(