CSE 341 - Programming Languages
Final exam - Winter 2015

Your Name:

```
Total  (max 110):
```

```
(for recording grades):
 1. (max 10)

 2. (max 10)

 3. (max 15)

 4. (max  8)

 5. (max  5)

 6. (max 10)

 7. (max 10)

 8. (max 10)

 9. (max  6)

10. (max 16)

11. (max  0)

12. (max 10)
```

You can bring a maximum of 2 single sided pages (or one double-sided page) of notes to the final. No laptops, tablets, or smart phones. The notes can be either your own notes, or printouts of materials from the class website. Please answer the problems on the exam paper — if you need extra space use the back of a page.

1. (10 points) Write a Prolog rule `doubles`. It should succeed if its argument is a list of numbers, such that the $n+1^{th}$ number is two times the n^{th} number. Use the clpr library. (Hint: remember that to write a constraint that uses the clpr library write it in curly brackets, e.g., `{X=Y}`.) Here are some examples of goals that should succeed:

```
doubles([]).
doubles([3]).
doubles([A,B,12,D,E]). /* should succeed with A=3.0, B=6.0, D=24.0, E=48.0 */
```

2. (10 points) Write an analogous `doubles?` function in Racket. It should succeed if its argument is a list of integers, such that the $n+1^{th}$ integer is two times the n^{th} integer. The Racket version doesn't need to allow for logic variables in the list, of course — it just takes a list of integers. You don't need to do any error checking. In analogy with the Prolog rule, `(doubles? '())` and `(doubles? '(10))` should both evaluate to `#t`.

3. (15 points) The parser provided for the OCTOPUS interpreter represents Racket lists using the constructor `OctoList` followed by a Haskell list of `OctoValue`s:

```
data OctoValue
      = OctoInt Int
      | OctoBool Bool
      | OctoSymbol String
      | OctoList [OctoValue]
        .....
```

Using this representation, `parse "()"` evaluates to `OctoList []`, and `parse "(1 2 3)"` evaluates to `OctoList [OctoInt 1, OctoInt 2, OctoInt 3]`.

Suppose instead that we represent OCTOPUS lists using explicit cons cells:

```
data OctoValue
      = OctoInt Int
      | OctoBool Bool
      | OctoSymbol String
      | OctoConsCell OctoValue OctoValue
      | OctoNil
        .....
```

(a) Using the new representation, what does `parse "()"` evaluate to?

What does `parse "(1 2 3)"` evaluate to?

(b) Write primitives for `octocons`, `octocar`, and `octocdr` (i.e., Haskell functions to implement the primitives for cons, car, and cdr), using the new representation for lists. Hint: here is the code for `octoplus` from the starter file. As with all of the Haskell functions to implement primitives, it takes a Haskell list of the arguments to the primitive.

```
getint (OctoInt i) = i
-- The octoplus function takes a list of OctoInts and adds them.
octoplus ints = OctoInt $ sum $ map getint ints
```

(c) A benefit of the new representation is that OCTOPUS could then handle improper lists. Draw a box-and-arrow representation of the improper list `(1 2 . 3)`.

(d) What should `parse "(1 2 . 3)"` evaluate to? (In other words, what gets printed if you type `parse "(1 2 . 3)"` at the ghci prompt after loading the OCTOPUS interpreter?)

4. (8 points) Consider the following Prolog rule `last`, which succeeds if the first argument is a list, and the second argument is the last element of that list.

```
last([X],X).
last([_|Xs],Y) :- last(Xs,Y).
```

What are all the answers returned to the following goals? (If there are an infinite number, give at least the first three.)

```
last([1,2,3],A).
```

```
last(As,10).
```

```
last([],A).
```

```
last([1,2,3],5).
```

Now suppose that we add a cut:

```
last_cut([X],X) :- !.
last_cut([_|Xs],Y) :- last_cut(Xs,Y).
```

What are all the answers in this case to the same goals?

```
last_cut([1,2,3],A).
```

```
last_cut(As,10).
```

```
last_cut([],A).
```

```
last_cut([1,2,3],5).
```

5. (5 points) Which of the following lists represent valid difference lists? For valid difference lists, what list do they represent?

```
[a,b,c]\[b,c]
```

```
[b,c]\[a,b,c]
```

```
T\T
```

```
[X|T]\T
```

```
[[1,2],[3,4],[5,6]]\[5,6]
```

6. (10 points) The `PosRational` class, which we used as an introductory Ruby example, includes a `+` method to add two positive rational numbers. However, this method doesn't interoperate correctly with integers. Write a modified version of the `+` method for `PosRational`, and also any additional new methods that you need, to make `r+3` and `3+r` work correctly for a postive rational instance `r`.

   Hint: here is the code for `PosRational` for `initialize` and `+`.

```
class PosRational

  def initialize(num,den=1)
    if num < 0 || den <= 0
      raise "PosRational received an inappropriate argument"
    end
    @num = num
    @den = den
    reduce
  end

  def + r
    ans = PosRational.new(@num,@den)
    ans.add(r)
    ans
  end
end
```

7. (10 points) Define a Ruby class `MyRange` that represents a range of integers, with an optional step. (Note that the `MyRange` class in the example notes didn't include a step.) Include the `Enumerable` mixin in your class. `MyRange` should include two methods: `initialize` and `each`. `initialize` should have three parameters: `first`, `last`, and `step`. `step` should be optional and default to 1. It can be negative, but you should raise an exception if it is 0. Here are some examples.

```
r = MyRange.new(1,5)
/* r.each should yield the values 1,2,3,4,5 */

r = MyRange.new(6,0,-2)
/* r.each should yield the values 6,4,2,0 */

r = MyRange.new(10,10)
/* r.each should yield the value 10 */

r = MyRange.new(6,10,-1)
/* r.each should yield no values */

r = MyRange.new(6,10,0)
/* this should raise an exception */
```

8. (10 points) Suppose that Java didn't overload method names, and allowed method typing to be contravariant in the argument types. Then suppose we have the following interface:

```
interface Octopus {
    public void test1(RectangularShape s);
    public void test2(ArrayList<Point> as);
}
```

For each of the method declarations in the following programs, say whether or not it results in a compile time error. You should have a total of 4 answers. (Hint: `Rectangle2D` is a subclass of `RectangularShape`.)

```
public class BabyOctopus1 implements Octopus {
    public void test1(Object s) {
        System.out.println("calling test1 in BabyOctopus1");
    }
    public void test2(ArrayList<Object> s) {
        System.out.println("calling test2 in BabyOctopus1");
    }
}
```

```
public class BabyOctopus2 implements Octopus {
    public void test1(Rectangle2D s) {
        System.out.println("calling test1 in BabyOctopus2");
    }
    public void test2(ArrayList<?> s) {
        System.out.println("calling test2 in BabyOctopus2");
    }
}
```

Finally, is this version of Java's type system, with contravariant typing for method arguments, sound? If it is sound, give an example method call that passes this version of the type system but that would result in a compile time error in standard Java. If it is not sound, give an example of a method call that passes type checking and that results in a runtime type error.

9. (6 points) Consider the following Java code fragments. In each case, does the code compile correctly? If so, does it execute without error, or is there an exception?

```
String[] a = new String[10];
Object[] b;
b = a;
b[0] = new Point(10,20);
```

```
Object[] a = new Object[10];
String[] b;
b = a;
b[0] = "squid!";
```

```
String[] a = new String[10];
Object[] b;
b = a;
b[0] = "squid!";
```

10. (16 points) Consider the following Ruby classes and mixins. (Feel free to tear this page out of the exam and not hand it in, if you don't have anything on it you want graded and if you want to have it side-by-side when looking at the expressions on the following page.)

```
class Class1
  def seacreatures
    ["octopus"] + others
  end
  def others
    ["squid"]
  end
end

module M1
  def seacreatures
    ["clam"] + super
  end
end

module M2
  def others
    ["oyster"]
  end
end

class Class2 < Class1
  include M1
end

class Class3 < Class1
  include M1, M2
end
```

Suppose we define the following variables:

```
c1 = Class1.new
c2 = Class2.new
c3 = Class3.new
```

What is the result of evaluating the following expressions?

```
c1.seacreatures

c2.seacreatures

c3.seacreatures

Class2.superclass

Class2.ancestors

Class3.ancestors

Class3.class

Class3.class.class
```

(Hint: `Object.ancestors` evaluates to `[Object, Kernel, BasicObject]`.)

11. (0 points) `equal?` is Ruby's idea of what the identity test should be called. Is this merely misguided, or in fact a sinister plot by the Ruby implementors to confuse generations of programmers as to the difference between object identity or object equality? Defend your answer. (Continue on the backs of the pages as needed.)

12. (10 points) True or false?

    (a) Suppose we had a dynamically typed version of Haskell called D-Haskell. Any program in normal Haskell that successfully compiles and executes would also successfully compile and execute in D-Haskell.

    (b) Any program in D-Haskell that successfully compiles and executes would also successfully compile and execute in normal Haskell.

    (c) In Ruby, if `x==y` evaluates to true, `x.equal?(y)` must evaluate to true as well; this is enforced by the language implementation.

    (d) Ruby blocks are not first-class citizens.

    (e) Java methods can be covariant in the return type.