Computer Science & Engineering 505 – Final

December 10, 1999
Open book & notes – 110 minutes – 10 points per question
90 points total

Name: _____

Please write any long answers on the back of the exam or on a separate piece of paper.

1. Suppose the following Miranda script has been loaded. (This script is type correct.)

```
fmap [] [] = []
fmap (f : fs) (x : xs) = f x : fmap fs xs

manyconsts = m 1
             where m i = my_const i : m (i+1)

my_const k x = k

double x = x+x
square x = x*x

my_abs x = x, if x>=0
         = -x, otherwise

mystery f x = f x : mystery f (f x)
```

   What is the result of evaluating the following Miranda expressions? (If the expression is followed by :: give its type.) If there is a compile or run time error, describe what the error is.

   (a) `double ::`

   (b) `my_const ::`

   (c) `my_const double ::`

   (d) `fmap ::`

   (e) `manyconsts ::`

   (f) `fmap manyconsts ::`

   (g) `fmap double [1,2,3] ::`

   (h) `fmap [double, square,abs] [1,2,3]`

   (i) `mystery ::`

   (j) `mystery double 1`

2. Suppose that we define a class `Animal` in Java, with a subclass `SeaCreature`. `SeaCreature` in turn has a subclass `Squid`. `Animal` and `SeaCreature` define a method `whenAttacked`, and `Animal` and `Squid` define a `startle` method.

```
public class Animal {
  public void whenAttacked() {
    this.startle();
    System.out.println("flee!!");
  }
  public void startle() {
    System.out.println("eek ...");
  }
}


public class SeaCreature extends Animal {
  public void whenAttacked() {
    System.out.println("swim away!");
    super.whenAttacked();
  }
}

public class Squid extends SeaCreature {
  public void startle() {
    System.out.println("how about a cloud of ink ...");
  }
}
```

What is printed when we invoke `whenAttacked` on an instance of Squid? On an instance of SeaCreature? On an instance of Animal?

3. Java does not allow two different methods with the same name and argument types but different return types. Consider changing the language to allow such overloading on the return type. Would there ever be ambiguous expressions? If so, give an example. Could the ambiguity be detected at compile time or only at run time? How could it be disambiguated?

   Also discuss the implications for progam maintainability. (Hint: are there Java programs that were correct but become incorrect after some modification?) Please write your answer on the back of this page.

4. Suppose we have a type `Fish` with a subtype `Salmon`. We also have a type `Fisher` that has a `catch` method with an argument of type `Fish`, and another type `SalmonFisher` that defines a `catch` method with an argument of type `Salmon`.

   We might sketch these classes in a Java-like language as follows:

```
class Fish
{ ...
}

class Salmon extends Fish
{ ...
}

class Fisher
{
    void catch(Fish f)
    { ...
    }
}

class SalmonFisher
{
    void catch(Salmon s)
    { ...
    }
}
```

   However, we aren't considering Java here, but rather languages that implement the contravariant or the covariant subtyping rules.

   What is the subtyping relation if any between `Fisher` and `SalmonFisher` under the contravariant typing rule?

   What is the subtyping relation if any between `Fisher` and `SalmonFisher` under the covariant typing rule?

   If one of the rules gives an incorrect answer, give an example of a program that is statically correct but that has a runtime type error; or one that will always execute without type errors, even though the rule says it is incorrect.

5. In Pizza we might define an interface `Sortable` that specifies a method `min` for finding the minimum of two values. The `min` method is defined for a sortable object, takes an object of the same type, and returns an object (yet again) of that type. Given this interface, we could then define a class `MyInteger` that implements it, and another class `MyString` that implements it as well. So we should be able to find the min of two MyInteger objects, and the min of two MyString objects; but not the min of a MyInteger and a MyString.

   This code should type check correctly:

```
MyInteger i,j,k;
MyString r,s,t;
...
k = i.min(j);
t = r.min(s);
```

   We might try defining this interface as follows:

```
interface Sortable {
  Sortable min(Sortable x)
}
```

   However, this isn't right. Give an example of a statement that type-checks correctly but that shouldn't. Now give a correct definition for `Sortable`.

6. Smalltalk has a pure object model, while Java uses a hybrid model. What are the advantages and disadvantages of each approach?

7. Consider the following Smalltalk class definition.

```
Object subclass: #Octopus
   instanceVariableNames: 'myblock'
   classVariableNames: ''
   poolDictionaries: ''

setValue: x
   myblock := [x].

setBlock: y
   myblock := y.

getValue
  ^ myblock value
```

What is the result of evaluating the following code? (The value in each case will be the value of the last `o getValue` expression.)

(a)
```
| o a |
a := 3.
o := Octopus new.
o setValue: a.
a := a+1.
o getValue
```

(b)
```
| o a b |
a := 3.
b := [a].
o := Octopus new.
o setBlock: b.
a := a+1.
o getValue
```

8. Suppose we modify CLP($\mathcal{R}$) by adding static type declarations. Are there programs that used to work correctly before adding type checking that now fail during the static type checking phase? Are there programs that used to fail at runtime (perhaps due to a type error) that now fail static type checking? Give examples.

9. Suppose that in some application of Cassowary it is important to be able to change the weight and/or strength of a constraint, and this operation should be fast and incremental. For example, we might want to change a `stay` constraint from weak to strong. In Cassowary as currently implemented this would need to be done by deleting the weak stay constraint and adding a strong stay constraint. Could this be done more efficiently? If so, how?

   Hints: Consider the case both of changing to or from a required to a non-required constraint, and also the case of changing the strength or weight of a constraint that is non-required both before and after the change. My answer to this question is quite short — if you find yourself describing an elaborate algorithm extension something has probably gone amiss.