

CMSC330 Spring 2009 Practice Problems 6

1. Programming languages

- Describe the difference between ad-hoc and parametric polymorphism.
- Describe the difference between starvation and deadlock.
- Describe how functional programming may be used to simulate OOP.
- Describe 2 differences between HTML and XML.
- Describe the difference between query languages and programming languages.

2. Polymorphism

Consider the following Java classes:

```
class A { public void a() { ... } }
class B extends A { public void b() { ... } }
class C extends B { public void c() { ... } }
```

Explain why the following code is or is not legal

- `int count(Set<A> s) { ... } ... count(new Set<A>);`
- `int count(Set<A> s) { ... } ... count(new Set);`
- `int count(Set s) { ... } ... count(new Set<A>);`
- `int count(Set<?> s) { ... } ... count(new Set<A>);`
- `int count(Set<? extends A> s) { ... } ... count(new Set);`
- `int count(Set<? extends B> s) { ... } ... count(new Set<A>);`
- `int count(Set<? extends B> s) { for (A x : s) x.a(); }`
- `int count(Set<? extends B> s) { for (C x : s) x.c(); }`
- `int count(Set<? super B> s) { for (A x : s) x.a(); }`
- `int count(Set<? super B> s) { for (C x : s) x.c(); }`

3. Multithreading

- Using Java Conditions, you must implement a synchronization construct called `MyBarrier`. A `MyBarrier` object is created with a certain value `n`. When a thread calls the method `enter()`, it enters the barrier and blocks until a total of `n` threads have entered the barrier. When the n^{th} threads enters the barrier, all the threads waiting at the barrier wake up and unblock, and the n^{th} thread continues without blocking. When a thread calls the method `reset()`, the barrier is reset so that it starts fresh in counting up to `n` (i.e., `n` more threads must enter the `MyBarrier`). You may start by modifying the following code fragment:

```
public class MyBarrier {
    public void MyBarrier (int n) { ... }
    public enter() { ... }
    public reset() { ... }
}
```

- Implement `MyBarrier` using Ruby monitors.
- Write a Ruby program that creates a barrier for 2 threads, then creates 2 threads that each print out "hello", enters the barrier, then prints out "goodbye".

4. Lambda calculus

Make all parentheses explicit in the following λ -expressions

- a. $\lambda x.xz \lambda y.xy$
- b. $(\lambda x.xz) \lambda y.w \lambda w.wyzx$
- c. $\lambda x.xy \lambda x.yx$

Find all free (unbound) variables in the following λ -expressions

- d. $\lambda x.x z \lambda y.x y$
- e. $(\lambda x. x z) \lambda y. w \lambda w. w y z x$
- f. $\lambda x. x y \lambda x. y x$

Apply β -reduction to the following λ -expressions as much as possible

- g. $(\lambda z.z) (\lambda y.y y) (\lambda x.x a)$
- h. $(\lambda z.z) (\lambda z.z z) (\lambda z.z y)$
- i. $(\lambda x.\lambda y.x y y) (\lambda a.a) b$
- j. $(\lambda x.\lambda y.x y y) (\lambda y.y) y$
- k. $(\lambda x.x x) (\lambda y.y x) z$
- l. $(\lambda x. (\lambda y. (x y)) y) z$
- m. $((\lambda x.x x) (\lambda y.y)) (\lambda y.y)$
- n. $((\lambda x. \lambda y.(x y))(\lambda y.y)) w$

Show that the following expression has multiple reduction sequences

- o. $(\lambda x.y) ((\lambda y.y y y) (\lambda x.x x x))$

5. Lambda calculus encodings

Prove the following using the appropriate λ -calculus encodings

- a. $\text{not (not true)} = \text{true}$
- b. $\text{or false true} = \text{true}$
- c. $\text{if false then } x \text{ else } y = y$
- d. $\text{succ } 2 = 3$
- e. $(* 1 3) = 3$
- f. $(+ 2 1) = 3$
- g. $(Y \text{ fact}) 2 = 2$ // you do not need to expand any operators except fact & Y

6. Operational semantics

Use operational semantics to determine the values of the following OCaml codes:

- a. 1
- b. + 3 7
- c. + 1 (+ 2 3)
- d. (fun x = 4) 5
- e. (fun x = + x 6) 7
- f. (fun x = (fun y = + y x)) 8 9

7. Markup languages

- a. Creating your own XML tags, write an XML document that organizes the following information: 1-hour test on Spanish Monday in Jiménez worth 15%. 1-hour test on Computers Tuesday in CSIC worth 10%. 30-minute test on Computers Friday in AVW worth 5%.

8. Garbage collection

Consider the following Java code.

```
Object a, b, c;  
public foo() {  
    a = new Object();    // object 1  
    b = new Object();    // object 2  
    c = new Object();    // object 3  
    a = b;  
    b = c;  
    c = a;  
}
```

- a. What object(s) are garbage when foo() returns? Explain why.
- b. Describe the difference between mark-and-sweep & stop-and-copy.