1. Programming languages
   a. Describe the difference between ad-hoc and parametric polymorphism.
   b. Describe the difference between starvation and deadlock.
   c. Describe how functional programming may be used to simulate OOP.
   d. Describe 2 differences between HTML and XML.
   e. 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
   a. int count(Set<A> s) { … } … count(new Set<A>);
   b. int count(Set<A> s) { … } … count(new Set<B>);
   c. int count(Set s) { … } … count(new Set<A>);
   d. int count(Set<?> s) { … } … count(new Set<A>);
   e. int count(Set<? extends A> s) { … } … count(new Set<B>);
   f. int count(Set<? extends B> s) { … } … count(new Set<A>);
   g. int count(Set<? extends B> s) { for (A x : s) x.a( ); }
   h. int count(Set<? extends B> s) { for (C x : s) x.c( ); }
   i. int count(Set<? super B> s) { for (A x : s) x.a( ); }
   j. int count(Set<? super B> s) { for (C x : s) x.c( ); }

3. Multithreading
   a. 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\textsuperscript{th} threads enters the barrier, all the threads waiting at the barrier wake up and unblock, and the n\textsuperscript{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( ) { … }
   }

   b. Implement MyBarrier using Ruby monitors.
   c. 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. \( \lambda x.z \lambda y.y \)
   b. \( (\lambda x.z) \lambda y.w \lambda w.wyz \)
   c. \( \lambda x.xy \lambda x.yx \)

   Find all free (unbound) variables in the following λ-expressions
   d. \( \lambda x.z \lambda yxy \)
   e. \( (\lambda x.z) \lambda y.w \lambda w.wyzx \)
   f. \( \lambda x.y \lambda x.yx \)

   Apply β-reduction to the following λ-expressions as much as possible
   g. \( (\lambda z.z) \lambda y.yy \)
   h. \( (\lambda z.z) (\lambda z.zz) (\lambda z.zy) \)
   i. \( (\lambda x.\lambda y.yx y) \lambda a.a \)
   j. \( (\lambda x.\lambda y.x y) \lambda x.yy \)
   k. \( (\lambda x.x) (\lambda y.yx) \)
   l. \( (\lambda z.\lambda y.(zy)) \lambda z.yyy \)
   m. \( ((\lambda x.x)(\lambda y.y)) \lambda y.y \)
   n. \( (((\lambda x.\lambda y.(zy)))(\lambda y.y)) \lambda x.x \)

   Show that the following expression has multiple reduction sequences
   o. \( (\lambda x.y) (\lambda y.yyy) (\lambda x.xx) \)

5. Lambda calculus encodings
   Prove the following using the appropriate λ-calculus encodings
   a. not (not true) = true
   b. or false true = true
   c. if false then x else y = y
   d. succ 2 = 3
   e. \( (* \ 1 \ 3) = 3 \)
   f. \( (+ \ 2 \ 1) = 3 \)
   g. \( (\text{Y fact}) \ 2 = 2 \) // 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.
   ```java
   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.