1. (20 points) Explain the difference between ordinary classes and “interfaces” in Java. Give an example of each.

2. (20 points) Give four ways in which Java is more object-oriented than C++. 

3. (40 points) Write a Java program that opens a file named on the command line, reads it, and writes it out to standard output with all blank lines removed.
4. (30 points) Write an Icon procedure `Generate(L)` that takes an arbitrary list L that may contain sublists, and sublists within sublists, and so on, and generates all the non-list elements out of L and any of its sublists.
5. (20 points) One of the big ideas in Icon was string scanning, where the syntax
   \[ s ? \text{expr} \]
   allows expression expr to analyze a string subject s and do arbitrary computations mixed in
   with the pattern matching code. The functions \text{tab()} and \text{move()} move a “scanning
   position” around within s, using absolute and relative indexing, respectively. Functions
   \text{any(c)} and \text{many(c)} check whether exactly one, or “one or more” matches of character set
   c occur at the current position, while \text{find(s)} and \text{upto(c)} generate all the matches (for
   string s, and character set c, respectively) from the current position forward. What is wrong
   with the following code? Write a fixed version.

   # find and produce (generate) all the words on a line
   procedure find_words(s)
      s ? {
         while word := tab(upto(' ')) do {
            suspend word
            move(1) # past the space
         }
      }
   end
6. (30 points) In Icon and Unicon, the unary question mark operator \( ?x \) returns a random element from \( x \). Write a Unicon class named \texttt{Niklaus} (yes, named after famous language inventor Niklaus Wirth) that contains a field named \texttt{gifts} and a method named \texttt{give_present()}. Initialize the \texttt{gifts} to a list containing the strings “doll” and “ball”, and write the body of \texttt{give_present()} to return a random element from \texttt{gifts}. Write a subclass of \texttt{Niklaus} named \texttt{Santa} with fields named \texttt{naughty} and \texttt{nice}. Initialize \texttt{naughty} and \texttt{nice} to an empty collection structure (list, set, or table; your choice). Give class \texttt{Santa} a method \texttt{know(name)} that checks whether \texttt{name} is in the naughty or the nice collection; if naughty, \texttt{know()} returns “bad”, if nice, \texttt{know()} returns “good”. If it is not on the list, randomly insert \texttt{name} into either the naughty or the nice collection, and then return “bad” or “good” appropriately. Give class \texttt{Santa} a method \texttt{give_present(name)} that returns the string “lump of coal” if \texttt{know(name)} returns “bad”, and returns a random “doll” or “ball” if \texttt{know(name)} returns “good”.