Exam 2 Information

The exam will be in class on Wednesday, 4 November. It will cover all material from Friday, 2 October through Friday, 30 October. The main topics we have covered in that time are:

- Control structures (recursion)
- Data types—basic types; type systems; polymorphism; strong typing; static typing; composite types
- Arrays; indexing, memory allocation, row- vs. column-major
- Haskell and Python basics
- Pointers
- Functions—arguments, parameters, runtime stack, value/reference passing, applicative/normal/lazy evaluation

(I may have missed a few—see slides.) Questions will be aimed at basic understanding of concepts and the ability to apply them in concrete examples. Question types will include:

- Short answer (may occasionally require writing simple code)
- True/false
- Multiple choice

You will not be asked to write whole programs; however, you have been exposed to a number of concepts through languages other than Java (in labs, in class) and you will be expected to recognize features of such languages when they were highlighted in class (e.g., use of pointers in C, function definition in Python and Haskell, list manipulation, etc.)

Here are examples of the kinds of questions that might be asked. This is not intended to be a sample exam; the topics covered below are not intended to be an exhaustive review. In particular, knowing the answers to all the questions below will not guarantee a good grade on the exam!

1. In Python, the following function is legal (sample run shown on the right):

```python
def f(a):
    if a == "dracula":
        return "frankenstein"
    elif a == 10:
        return 20
    else: return False
```

```python
>>> from exam2rev import *
>>> f(10)
20
>>> f("dracula")
"frankenstein"
```

HANDED OUT ON 30 OCTOBER 2015

Handout 16
True or False: This demonstrates the fact that Python is statically typed.

2. What are the final values of the variables `boo` and `ghost` in the following C code?

```c
double boo = 42.0;
double ghost = boo;
double *phantom = &ghost;
*phantom = 21.0;
```

3. Consider the following nested JavaScript function definitions:

```javascript
function mummy() {
    function wolfman() {
        function vampire() {
            ...
        }
    }
    function chupacabra() {
        ...
    }
    ...
}
```

Which of the following calling sequences are legal (there could be none, one, or several):

(a) `mummy` calls `vampire` calls `wolfman`
(b) `chupacabra` calls `chupacabra` calls `wolfman`
(c) `wolfman` calls `mummy` calls `mummy` calls `wolfman`
(d) `mummy` calls `chupacabra` calls `vampire`

4. In Haskell, what is the value of the expression:

```
(take 1 ["bat","wolf","spider"] ++ (drop 2 ["igor","vlad","boris"]))
```

5. Which of the following Java functions correctly finds the sum of the digits in a nonnegative integer named `eeek`? For instance, `sum(354) = 3+5+4=12` and `sum(10223) = 1+0+2+2+3=8`.
6. In Java, an object may be considered to be of more than one type; for instance, an instance of a subclass named Kong can also be considered as an instance of the parent class Gorilla. What is the term for the ability of a variable to be simultaneously considered as two different types?

7. Suppose a Java array named fang is declared as follows:

   int fang[][] = new int[5][3];

   Assuming an int takes 4 bytes and assuming that the array is stored in row-major order, write the formula for calculating the address of element fang[r][c], where r and c are valid indices and the base address of the array is 0.

8. Define “orthogonality” in a programming language.

9. In Scott's book, the following phrase appears: “It is sometimes argued that iteration is more efficient than recursion. It is more accurate to say that naive implementation of iteration is usually more efficient than naive implementation of recursion.” What type of recursion can be handled as efficiently as iteration?

10. In Fortran, arrays are stored in column-major order; in C and Java they are stored in row-major order. Does it make any difference to the programmer which method is used? Explain.

11. What is currying? Give an example.

12. What are the two principle purposes of “types” in a language (according to the textbook)?

13. Give examples of primitive types; give examples of composite types.

14. What is an “enumerated type”? Give an example.

15. In Haskell interactive mode, what is printed by the following set of statements?

```
Prelude> let creature black lagoon = head black ++ (tail lagoon)
Prelude> let black = [[1,2,3],[4,5,6]]
Prelude> let lagoon = [7,8,9]
Prelude> creature black lagoon
```
16. If you have tried to track down precise definitions of the terms “strongly typed” and “weakly typed” you have probably come away frustrated. Scott defines a “strongly typed language” as one that “prohibits . . . the application of any operation to any object that is not intended to support that operation.” Give an example of such an operation and such an object, using only standard primitive types.

17. [From the Slack postings in Wednesday’s class.]

The above is not a full review; this is what I had time to prepare by Friday morning. Please come to Monday’s class prepared to ask questions about material on the exam. I will also try to have the slides collected into one file and perhaps have some additional slides expanding on topics you have mentioned in Wednesday’s class.