CMSI 386

Final Exam

The test is open-everything with the sole limitation that you neither solicit nor give help while the exam is in progress.

Problem	You got	Out of
1		10
2		10
3		10
4		10
5		10
6		10
7		10
8		10
9		10
10		10
TOTAL		100

1. Show the output of the following code fragment under the following four conditions: (a) pass by value, (b) pass by value-result, (c) pass by reference, and (d) pass-by-name.

```
x = 1
y = [2, 3, 4]
function f(a, b) {a--; b = x - 1}
f(x, y[x])
print x, y[0], y[1], y[2]
```

2. What does this script print under (a) static scope rules and (b) dynamic scope rules?

```
var x = 1
function h() {var x = 0; return g()}
function f() {return x}
function g() {var x = 4; return f()}
print f() - h() + x
```

3. Show the output of the following, assuming dynamic scope and (a) deep binding, and (b) shallow binding.

```
function f(a) {
  var x = a - 1
  function g() {
    print x - 17
  }
  h(g)
}
function h(p) {
  var x = 30
  p()
}
f(18)
```

4. Write a JavaScript "module" (i.e., use the famous module pattern employing a closure) that contains (1) a (private) string, initialized to "MI", (2) a function which retrieves the value of this string, and (3) four other methods, each of which changes the string to another string according to the following:
    - If the current string ends in an "I", add "U". e.g. "MI" → "MIU".
    - Append the second through last characters of the string onto itself, e.g. "MIIU" → "MIIUIIU".
    - Replace any occurrence of "III" with "U", e.g. "MUIIIU" → "MUUU".
    - Remove any "UU", e.g. "MIUUUI" → "MIUI".

    Remember, the point is that the string inside the module can not be changed by any operation other than the four defined above.

5. Write a Python class for students that have an id, name, birthday, and transcript. The transcript is a `dict`, indexed by semester name, whose values are `dicts` mapping course numbers to grades. Supply methods to read the id, read and write the name, read and write the birthday, get the grade for a given course (in any semester), and add an item to the transcript. Did you encapsulate (i.e. hide) the properties inside the class so they were protected from direct access from outside the class? Why or why not?

6. Remember that problem you did for homework regarding Rubin's claim to have found a chunk of code for which a goto was suitable? Write a function to find the index of the first all-zero row of a two-dimensional array (technically an array of arrays) in JavaScript. Use JavaScript array methods such as `every`, `some`, `forEach`, etc.

7. Write a code fragment in either Go or C++ that fills an array, indexed from 0 through 9, such that slot *i* of the array contains a function (or pointer to a function if your language demands it) that divides its argument by the square of *i*.

8. Write a tail recursive function that takes in an array, and returns a new array just like the old one, except with the values at even-numbered indexes removed. (It is okay to use a "helper" that is tail-recursive) For example:

   ```
   withoutEvens([4,6,true,null,2.3,"xyz"]) => [6,null,"xyz"]
   ```

   You can use any language you like, as long as *I* know it. *Consider efficiency for full credit*.

9. Here's an operator chart from some unnamed language. Operators are listed in decreasing precedence (highest precedence operators are at the top of the table; lowest precedence operators are at the bottom). Operators listed on the same line have the same precedence.

Operator(s)	**Assoc**	**Arity**	**Fixity**
§ #		1	Prefix
∀ ⊥		1	Postfix
% • ¥	L	2	Infix
⊗ ∇ ⇒	R	2	Infix

Draw the expression tree for the following

```
# # A ⇒ # B ∇ C ¥ D % E ∇ F % G • H ⊥ ∀ ⊗ § I ∀
```

10. Suppose you were designing a programming language (yes it could happen) and you chose the following types: Null, Boolean, Number, String, Array, Dictionary, and Function. Suppose you wanted to make the language **100% weakly typed**. In order to do this you would have to come up with a rule for implicitly converting any expression of any type into a reasonable "equivalent value" in another type, essentially completing this table:

To→ From⤵	**Bool**	**Num**	**Str**	**Arr**	**Dict**	**Fun**
Null	false	0	""	[]	{}	
Bool		false→0 true→1				
Num	0→false else true					
Str	""→false else true					
Arr	[]→false else true					
Dict	{}→false else true					
Fun	true					

Explain how you would complete this table. You can write some things in the table and then complete your answer with prose in the space below or on the back of this page; for example, "to coerce anything at all to a function, blah blah blah."

EXTRA CREDIT (2 points but the whole answer must be *exactly what I am looking for*): Is JavaScript 100% weakly typed? About what percentage is it? Why, exactly, is it not?