Midterm #1 Free Response Grading Rubric

Overview
In general, each grading rubric follows the following policy:

- If the code runs correctly, even if it is not an “ideal” answer, it is worth **15 points**.
- If it is clear that “I don’t know” or “idk” is written as the answer they wish to have graded, give them **3 points**.
  - If “idk” **and** an answer appears **and** it is unclear that “idk” is their absolute choice for what should be graded, grade the answer and ignore the “idk”.
- Otherwise:
  - For code that needs only a minor syntax change, subtract 2 points for each category of error. Examples include:
    - **-2 points** if they return “true” (a string) instead of true (Boolean)
    - **-2 points** if they forget ()s around their Boolean inside of an if-statement
    - **-2 points** if they forget to specify the parameter but use it as though it was specified
    - **-2 points** if they miss one specific case (eg: returns on <10 and >10, but forgets about the ==10 case).
    - **...generally, everything that requires changing only a few characters should get -2 points. Most notable exception to this rule is forgetting to index into an array data type; a == 4 vs a[i] == 4 is major.**
  - For code that has a single major structural problem, subtract 8 points for the major error. Examples include:
    - Returning only true or only false, forgetting half of the return values
    - Returning within for-loops
    - Other structural problems, often explained in the question-specific stuff
- If the code does not relate to the problem in a major way (eg: using a loop with an array when there is no array in the problem), it should be awarded **0 points**.
  - A solution scoring 13 requires only a few characters changed
  - A solution scoring 7 requires only a single line added/removed/changed
  - A solution scoring 3 requires two minor changes plus a line added/removed
  - *Just because the solution given has a “correct piece” does not grant it any points.*

Some final notes on all of the problems

- For each specific type of minor/major problem, count each individual problem type only once. Returning both ”true” and ”false” as strings is only -2.
- Ignore unnecessary code and pretend it does not exist. If an extra variable is defined and is never used, that’s fine.

---

**check() A: FR1  B: FR2  C: FR3  D: FR1**

Write the `check()` function that was defined in the previous question. Remember that `check()` takes in two parameters and returns a value. Your answer must include the full function, not just what is contained inside of the function.

**Ideal Answer**

```javascript
function check(a, b)
{
    if ( a == b ) { return true; }
}```
else { return false; }
}

Noteworthy Alternative Answers
function check(a, b)
{
    return ( a == b );          // Clever, and correct
}

function check(a, b)
{
    if ( a === b ) { return true; } // Triple equals is OK
    return false;                  // (single equal is not)
}

function check(a, b)
{
    if ( a == b ) { return true; }
    if ( a != b ) { return false; }
}

Grading Rubric
For solutions close to a correct solution
For minor coding errors, -2 points. For this question, this includes:
- Using a single-equals in the conditional, \( a = b \)
- Returning "true" as a string, instead of the Boolean value
- Swapping the logic of true/false (eg: returning false on equals)

For each minor/major problem, count each individual problem only once. Returning both
"true" and "false" as strings is only -2.

For major structural problems, -8 points. For this problem, this includes:
- Returning only true or only false, excluding the other one

For any code that does not structurally solve the problem, 0 points.
- This would include code that calls check() within itself, code that has any sort of a for-
  loop treating a parameter like an array, or other unrelated structure.
- This also includes the use of any loops

checkAccess() A: FR2, B: FR3, C: FR1, D: FR3
Write a JavaScript function called checkAccess() that takes in two location objects as
parameters, university_location and user_location, and returns true if and only if the
user's location is within 10 miles of the university's location. Otherwise, the function must return
false. In your function, you must use the findDistance() function defined at the top of this
page to find the distance.

Ideal Answer
function checkAccess(university_location, user_location)
{
    if ( findDistance(university_location, user_location) <= 10 )
Noteworthy Alternative Answers

function checkAccess(a, b)  // Parameter names can be whatever
{
  return ( findDistance(a, b) < 10 );  // Clever, and correct
     // Allow for both < 10 and <= 10
}

function checkAccess(a, b)
{
  if ( findDistance(a, b) < 10 ) { return true; }
  return false;  // Would already return if true before here
}

Grading Rubric

If their code is correct and would run correctly, **full 15 points**.
- Ignore any findDistance() function that is re-written, if it appears outside of the checkAccess() function… it was not necessary to re-print it.

For solutions close to a correct solution

For minor coding errors, -2 points. For this question, this includes:
- Using an incorrect comparer in the conditional, eg: `(fundDistance(...) == 10)`
- Returning "true" as a string, instead of the Boolean value
- Swapping the logic of true/false (eg: returning false on equals)
- Comparing some other number than 10

For each minor/major problem, count each individual problem only once. Returning both "true" and "false" as strings is only -2.

For major structural problems, -8 points. For this problem, this includes:
- Returning only true or only false, excluding the other one
- Not checking findDistance(...) with 10
- Using findDistance as a variable instead of a function

For any code that does not structurally solve the problem, **0 points**.
- This would include code that calls checkAccess() within itself, code that has any sort of a for-loop treating a parameter like an array, or other unrelated structure.

---

red()  A: FR3, B: FR1, C: FR2, D: FR2

Write a JavaScript function called `red()` that takes in one input parameter, an array of card suits (*not the full card*), and returns the number of red cards in the hand. A card is considered red if the suit is either a diamond ("D") or a heart ("H"). As an example, the array `["D", "H", "C"]` contains two red cards.

**Ideal Answer**
function red(suits) {
    var ct = 0;
    for (var i = 0; i < suits.length; i++) {
        if (suits[i] == "H" || suits[i] == "D") {
            ct++;
        }
    }
    return ct;
}

Noteworthy Alternative Answers
function red(a)  // Parameter name doesn’t matter
{
    var ct = 0;
    var i = 0;
    while (i < a.length) {  // while-loops are okay if done correctly
        if (a[i] != "C" || a[i] != "S") {
            ct++;  // ^: Awkward, but can check for not a black card
        }
        i++;  // necessary for a while-loop
    }
    return ct;
}

function red(a) {
    // if you really want a one-line solution, way beyond CS 105:
    return a.filter(function(x) { return (x == "H" || x == "D"); }).length;
}

Grading Rubric
For solutions close to a correct solution
For minor coding errors, -2 points. For this question, this includes:
- Missing important syntax (eg: commas vs. semi-colons in for-loop)
- Using OR without referring to the parameter a second time, (a[i] == "H" || a[i] == "D")
- Missing i++

For major structural problems, -8 points. For this problem, this includes:
- Comparing suits array without indexing into them for their string suits == "H"
- Returning within the for-loop
- Checking for only one red suit

For each minor/major problem, count each individual problem only once. Checking both suits == "H" and suits == "D" as strings is only -2.

For any code that does not structurally solve the problem, 0 points.