1. Write code to put 10 random integers in the range 0…100 into an array.

```javascript
var grades = new Array;
for (var i=0; i<10; i++)
    grades[i] = Math.round(100*Math.random());
```

Math.random() returns a number from 0 up to, but not including 1. When you multiply by 100, you get a number from 0…99.9999… Then rounding gives an integer 0…100.

[12 points total]

2. Prompt the user to enter positive numbers (> 0) --- keep prompting till the user enters a number less than or equal to 0. Store all of the positive numbers entered in an array. Then display (in a single HTML text line) only those elements of the array that are even.

```javascript
var userInput = 1*prompt("Enter a number: ");
var count = 0;
var grade = new Array;
while (userInput > 0) {
    grade[count] = userInput;
    count++;
    userInput = 1*prompt("Enter a number: ");
}
var s = "";
for (var i=0; i<grades.length; i++)
    if (grades[i]%2 == 0)  // Test to see if even.
        { s = s + " " + grades[i]; }
document.write(s);
```

[16 points total]

3. Write code that takes the array created in the exercise above and creates another array containing only those elements of the original array that are in the range 25 … 75. Display the elements of this second derived array in a single HTML line. Then compute and display the average of the numbers in this derived array (using an alert box).

```javascript
var newGrades=new Array;  // Will hold grades in 25…75
var newGradesIndex=0;
for (var i=0; i<grades.length; i++)
    if (grades[i]>=25 && grades[i]<=75)
        { newGrades[newGradesIndex]=grades[i];
        newGradesIndex++;}
// newGrades now contains elements from grades with values in 25…75.
var sum=0;
for (var i=0; i<newGrades.length; i++)
    { sum = sum + newGrades[i];}
var average = sum/newGrades.length;
```
For the following four exercises, assume you are given a function \( p(x) \) that takes a parameter \( x \) and returns true if \( x \) has property \( p \), and false if \( x \) does not have property \( p \). It doesn't matter what property \( p \) is. Assume that the function \( p \) works for any \( x \).

4. Write a function \( \text{allP}(a) \) that takes the array \( a \) as its parameter and returns true if all of the elements of \( a \) have property \( p \). It returns false if not all the elements of \( a \) have the property. If an array has zero elements, then \( \text{allP}(a) \) is true. (Think about why this is the case. For example, if you turned in no assignments in a course, would it be truthful to tell your parents, "Every assignment I turned in got an A."?)

    // Do all of the elements of array \( a \) have property \( p \)?
    function \( \text{allP}(a) \)
    {
        for (var \( i=0; i<\text{a.length}; i++ \))
            if (!\( p(\text{a}[i]) \)) return false;   // Found an element that does not have property.
        return true;  // Survived the loop.
    }

[10 points total]

5. Write a function \( \text{someP}(a) \) that takes the array \( a \) as its parameter and returns true if some (one or more) of the elements of \( a \) have property \( p \). It returns false if none of the elements of \( a \) have property \( p \). What should \( \text{someP}(a) \) be if \( a \) has no elements?

    // Do one or more of the elements of array \( a \) have property \( p \)?
    function \( \text{someP}(a) \)
    {
        for (var \( i=0; i<\text{a.length}; i++ \))
            if (\( p(\text{a}[i]) \)) return true;   // Found an element that has the property.
        return false;  // Made it through the loop without finding an element that has the property.
    }

[10 points total]

6. Write a function \( \text{countP}(a) \) that takes the array \( a \) as its parameter and returns the number of elements of \( a \) that have property \( p \). What should be returned if the array has no elements?

    // How many of the elements of array \( a \) have property \( p \)?
    function \( \text{countP}(a) \)
    {
        var \( \text{count}=0; \)
        for (var \( i=0; i<\text{a.length}; i++ \))
            if (\( p(\text{a}[i]) \)) \( \text{count}++ \);   // Found an element that has the property.
        return \( \text{count} \);
    }

[10 points total]
7. Write a function subP(a) that takes the array a as its parameter and returns another array containing only those elements of a that have property p. What should allP(subP(a)) return? Will this work properly even if none of the elements of a have property p?

```javascript
// Create and return an array containing only those elements of p // that have property p.
function subP(a)
{
  var res=new Array;
  var resIndex=0;
  for (var i=0;i<a.length;i++)
  {
    if (p(a[i])) // Found an element that has the property.
    {
      res[resIndex]=a[i]; // Add found element to result array.
      resIndex++; // Update index.
    }
  }
  return res;
}
```

[15 points total]

8. Fill in the code to complete the following function which determines if an array contains three or more consecutive ones.

```javascript
function threeOnesInARow(ar)
{
  // Return true if the array ar contains three or more ones in a row;
  // Return false if the array does not contain at least three ones in a row.
  // Parameter: ar, an array to be searched.
  // Restrictions: ar must be an array.
  // Errors checked for: none.
  var goal = 3;
  var oneCounter = 0;
  for (var i=0; i<ar.length; i++)
  {
    if (ar[i] == 1)
    {
      oneCounter++; // Found a one.
    }
    else
    {
      oneCounter = 0; // Found something other than a one.
    }
    if (oneCounter >= goal){return true;} // Found enough ones.
  }
  // Survived to this point without returning. There must be no run of ones.
  return false;
}
```

[15 points total]

9. Write a recursive function power(x,n), that returns the value of \(x^n\) and relies on the following relations:

\[
\begin{align*}
x^n &= 1, & \text{if } n &= 0, \\
x^n &= x \cdot x^{n-1}, & \text{if } n &\geq 1
\end{align*}
\]
\[ x^n = \frac{1}{x} \times x^{n+1}, \text{ if } n \leq -1 \]

The function should display an appropriate error message (and should return a value of 0) if either \( x \) is 0, or if \( n \) is not an integer.

```javascript
function power(x, n)
{
    // First check if x is 0
    if (x == 0) {
        alert("base is zero!");
        return 0;
    }

    // Then verify that n is an integer
    if (isNaN(n) || (Math.floor(n) != n))
    {
        alert("power is not an integer!");
        return 0;
    }

    // Returns 1 if n is 0
    if (n == 0)
        return 1;

    // if n >= 1, use first recursive relation
    if (n >= 1)
    {
        return (x * power(x, n-1));
    }

    // if n <= -1, use second recursive relation
    if (n <= -1)
    {
        return (1/x * power(x, n+1));
    }
}
```

[15 points total]