Introduction to (Web) Programming — **Problem Set #1**
Attempt seriously by Thursday, March 7, before lecture
Submit hard copy on Monday, March 11, at start of lecture

Instructions

- Do the problem set entirely on your own. Questions should be directed to the instructor and no one else.
- Write your name on the top of the first page (this page).
- When asked, provide clear, *concise* explanations.
- *Handwrite* solutions in **black ink**. If you have trouble writing legibly (I sympathize!) then you may *staple* one or more typed pages (printed in *black ink* on *otherwise blank paper*) to your problem set. Some problems require you to annotate the problem itself (e.g., to circle parts of some code).

---

1 programming is strange

For each of the following expressions, indicate to what the expression evaluates and explain, briefly, why that result might be surprising to someone first learning to program.

a. `'47' === 47`

b. `'Thanos' > 'ultron'`

c. `'MAGIC'.charCodeAt(1) === 65`

d. `parseInt('my favorite number') === NaN`

2 expressions and statements

Consider the following code fragment:

```
1   let n = parseInt(prompt('Enter a number: '));
2   let t = 0;
3   let i = 0;
4   while (i !== n) {
5       if (i % 2 === 1) {
6           t += 2;
7       } else {
8           t += 2;
9       }
10      i++;
11  }
12  alert('Final result: ' + t);
```

a. Circle all of the Boolean-valued expressions.

b. Circle the line numbers that correspond to the first line of each *compound* statement.

c. In a clear and concise sentence, what value does the `alert` display to the user (in terms of `n`)?

d. Would you describe this as an efficient way to achieve that computation? Explain.

e. Would you describe this code as *robust*? Explain — and try to be specific.

3 binary numbers

a. Express **46** in binary. Show your work.

b. Express the binary number **1001101** in base 10. Show your work.

c. Express the binary number **11110001** in base 10. Show your work.

d. Express **223** in binary. Show your work.

4 error!

Consider the following function:

```
function findFirstA(s) {
    let k = 1;
    let aPos = null;
    while (aPos !== null || k <= s.length) {
        let c = s.charAt(k);
        c = c.toLowerCase();
        if (c === 'A') {
            aPos = k;
            k++;
        }
    }
    return posA;
}
```

It is intended to return the index in string `s` at which the letter "a" (lowercase or uppercase) first occurs. It should return `null` if no "a" exists in the string. Examples of how it *should* behave:

```
> findFirstA('Alexander Hamilton')
0
> findFirstA('Fun fact')
5
> findFirstA('Nothing to see here...')
null
```

Identify errors by circling (or highlighting) specific parts of code. For each, briefly explain its effect and how to fix it.

5 evaluating expressions

Assume the following lines have just been executed in order:

```
let a = 22;
let b = 'true';
let c = prompt('Shall we play a game?');
let d = 'side';
let e = d.length;
let f = e > a;
let g = d.charAt(0) === 's';
let h = !(d.charAt(1) === 'i');
let i = 'Up' + d;
let j = i < d;
let k = a / 2;
let l = a % 2;
let m = k < a;
let n = l === 1;
let p = Math.random();
let q = p !== p;
let r = p === Math.random();
let s = isNaN(p);
```

Indicate the value of each of the variables below by ***circling*** exactly one of **t** (`true`), **f** (`false`) *NB* ("not Boolean" — i.e., the value is not of Boolean type) or *?* (cannot determine the answer with information provided). In the space to the right of your answer, briefly explain your reasoning.

var	*value*				*reason*
a	**t**	**f**	*NB*	*?*	
b	**t**	**f**	*NB*	*?*	
c	**t**	**f**	*NB*	*?*	
d	**t**	**f**	*NB*	*?*	
e	**t**	**f**	*NB*	*?*	
f	**t**	**f**	*NB*	*?*	
g	**t**	**f**	*NB*	*?*	
h	**t**	**f**	*NB*	*?*	
i	**t**	**f**	*NB*	*?*	
j	**t**	**f**	*NB*	*?*	
k	**t**	**f**	*NB*	*?*	
l	**t**	**f**	*NB*	*?*	
m	**t**	**f**	*NB*	*?*	
n	**t**	**f**	*NB*	*?*	
p	**t**	**f**	*NB*	*?*	
q	**t**	**f**	*NB*	*?*	
r	**t**	**f**	*NB*	*?*	
s	**t**	**f**	*NB*	*?*	

6 if/elif/else

Consider the following function:

```
function multiway(s) {
    let n = s.length;
    let a = s.charCodeAt(0);
    let b = s.charCodeAt(1);
    let c = s.charCodeAt(2);
    let z = 63;    // line A
    if ((n > 2) && (a < b) && (b < c)) {
        z -= 2;    // line B
    } else if ((n > 1) && (a > b)) {
        z -= 4;    // line C
    } else if ((n < 4) || (a === b)) {
        z -= 8;    // line D
    } else {
        z -= 16;   // line E
    }
    z = Math.floor(z / 2);   // line F
    return z;
}
```

a. Evaluate each of the following expressions (i.e., show what value the function would return) and indicate which of the lines (**A** – **F**) would be executed on the way toward returning the final result.

`multiway('')`

`multiway('ACT')`

`multiway('zoo')`

`multiway('Peru')`

`multiway('oodles')`

---

b. What kinds of strings (in terms of length) cause `multiway` to return `23` when passed in as arguments? Give a specific example of a such a string.

c. To evaluate these problems, you are welcome to use an ASCII chart. However, explain why you should not need such a chart to evaluate `multiway` by hand for the kinds of arguments it is being passed in the above problems.

7 arithmetic function

Consider the following function:

```
1  function co(m) {
2      let d = 0;
3      let n = m;
4      while (n > 0) {       // <---
5          if (n % 10 === 0) {
6              d++;
7          }
8          n = Math.floor(n / 10);
9      }
10     return d;
11 }
```

a. What does `co(5)` return?

b. What does `co(20)` return?

c. Suppose `co(70409)` is evaluated. Indicate the values of `n` and `d` the first four times line 4 is executed.

#	n	d
1		
2		
3		
4		

d. Write a one-sentence description what function `co` returns.

8 string function

Consider the following function:

```
1  function ghost(s) {
2      let b = true;
3      let k = s.length - 1;   // <---
4      let i = 0;
5      while (i < k) {
6          let c = s.charAt(i);
7          let d = s.charAt(i + 1);
8          if (c >= d) {   // <---
9              b = false;
10         }
11         i++;
12     }
13     return b;
14 }
```

a. What does `ghost('I')` return?

b. What does `ghost('if')` return?

c. Suppose `ghost('Hello')` is evaluated. Indicate the values of `b`, `c`, `d`, and `i` each time line 8 is executed. (*Leave extra rows, if any, blank.*)

#	b	c	d	i
1				
2				
3				
4				
5				
6				

d. Provide a string of length 5 for which `ghost` would return `true`.

e. Write a one-sentence description for what function `ghost` returns.

f. Would you describe this as an efficient way to achieve that computation? Explain.

g. Explain the effect of replacing line 3 with the following:

```
let k = s.length;
```

9 functional abstraction

Suppose we have three functions `fee`, `see`, and `tee` where `fee` and `see` are defined as follows:

```
function see(n) {
    let b = false;
    if (n % 3 === 0) {
        b = tee(n, n);
    } else {
        b = tee(n, n + 1);
    }
    return b;
}

function fee(u, v) {
    let z = 0;
    let j = u;
    while (j < v) {
        if (see(j)) {
            z++;
        }
        j++;
    }
    return z;
}
```

and `tee` is defined (but not shown here) and runs without error, takes two arguments, always returns a Boolean value, *and*, if its arguments are equal, it returns `True`. (We have no idea what it returns if its arguments are not equal.) Suppose `fee(5, 13)` is evaluated. What are the smallest and largest values that it might return? Explain your reasoning.

---

10 the perils of forgiveness

Give two examples of how JavaScript is a "forgiving" programming language and how that might make it more difficult to learn.