PIC 40A: Homework 2 (due 04/16 at 5pm)

Just for this homework on pure JavaScript, no submission to the PIC server is necessary! We will be creating some of the functions you will use for your merchandise page and login page.

In this assignment you will make one file called `hw2.js` and all you need to do is submit this file to Gradescope before the deadline. Start by opening `hw2.js`.

1. We will define a list of functions in order to perform the following computations. We will heavily rely on the mathematics functions in javascript for the first question. Find a reference for these functions at https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math.

   (a) First, let u_1 and u_2 be two random numbers in $(0, 1]$. Then, we define s as

   $$s = \sqrt{-2\ln(u_1)} \cdot \cos(2\pi u_2)$$

   For parameters $a, b \in \mathbb{R}$, define a function called `computation1` which returns

   $$\text{computation1}(a, b) = a + s * b.$$

   In particular, its function comment reads as follows.

   ```
   /**
   Returns a + s*b following the formula detailed above.

   @return {number} a + s*b
   */
   ```

   (b) Let $a, b, c, d \in \mathbb{R}$ be parameters. Define the following quantities

   $$e = \left(a - \frac{1}{2}b^2\right) \cdot c \qquad\qquad f = \sqrt{c} \cdot b.$$

   Define a function called `computation2` which returns

   $$\text{computation2}(a, b, c, d) = d \cdot \exp(\text{computation1}(e, f))$$

   In particular, its function comment reads as follows.

   ```
   /**
   Returns d * exp(computation1(e,f)) following the formula detailed above.

   @return {number} d * exp(computation1(e,f))
   */
   ```

(c) Let $a, b, c, d, K \in \mathbb{R}$ and $n \in \mathbb{N}$ be parameters. Since the return of `computation1` is a random number, the return of `computation2` is also random number. Let r_i be the random numbers defined by

$$r_i = \text{computation2}(a, b, c, d)$$

for $0 \leq i < n$ (note that there are exactly n numbers r_i). Define a function called `computation3` which returns

$$\frac{\exp(-a \cdot c)}{n} \sum_{i=0}^{n-1} \max(r_i - K, 0)$$

if $d > 0$ and 0 else. In particular, its function comment reads as follows.

```
/**
Returns \frac{exp(-a \cdot c)}{n} \sum_{i=0}^{n-1} \max(r_i -K,0) following the
     formula detailed above.

@return {number} 0 if d is less or equal to 0 and \frac{exp(-a \cdot c)}{n} \
    sum_{i=0}^{n-1} \max(r_i -K,0) else
*/
```

For the next questions, consider the following definition. This definition is not as general as possible: https://stackoverflow.com/questions/1969232/.

Definition. A **cookie** is a special type of `string`.

It has the form `"name1=value1; name2=value2; ...; lastName=lastValue"`.

The `names` and `values` can be any **non-empty** sequence of ASCII characters which are:

(a) alphanumeric characters: `a`, `b`, `c`, ..., `z`, `A`, `B`, `C`, ..., `Z`, `0`, `1`, `2`, ..., `9`, or

(b) a character appearing in the following string: `"!#%'*+-.^_`|~"`.

(c) `=` is allowed in a `value`, but not a `name`.

In particular, the following characters are **not allowed**:

- white space
- `,` (commas)
- `;` (semicolons)

For example, the following are examples of cookies:

- `"first_name=Sarah; last_name=Burnett; username=burnett"`

- `"username=burnett; first_name=Sarah; last_name=Burnett"`

- `"_ga=GA1.2.34.56; dwf_sg_task_complete=False; lux_uid=888; _gid=GA1.2.88.88"`
- `"__stripe_mid=c4d6a-723-3ee-54d-640e5af; csrftoken=Kger31Gtcvyt%2F2ILWuQJoJ"`

The following is **not** a cookie `"name=Sarah Burnett; position=PIC;assistant,adjunct"` because of the space between `Sarah` and `Burnett`, as well as the semicolon and comma in `PIC; assistant,adjunct`.

Finally, here are the actual questions...

2. Define a function called `extract_username` whose function comment reads as follows.

```
/**
This function extracts from a given cookie
the 'value' corresponding to the 'name' "username".

For example, both of the following function calls return "bur=nett":
. extract_username("first_name=Sarah; last_name=Burnett; username=bur=nett");
. extract_username("username=bur=nett; first_name=Sarah; last_name=Burnett");

If the given cookie has no 'name' called "username",
then the function returns the empty string.

For example, we have
. extract_username("common_error=Sara; " +
                   "another_one=Burnet; another=Sarah_Brunette") === "";

@param  {string} cookie : The cookie to extract information from.
@return {string} The 'value' corresponding to the 'name' "username";
                 the empty string if "username" is not a 'name'.
*/
```

if “username” return `value-> username` else ‘ ’

3. Define a function called `validate_username` whose function comment reads as follows.

```
/**
 This function validates a username.
 A string should be printed to the console after the username is evaluated.
 It'll be determined acceptable or relavent username errors will be printed.

 @param  {string} username : The username to validate.
 */
```

Upon giving this function a username up to three things should happen.

(a) The specified username should be checked to see if it is of a desired form:

- at least `5` characters,
- at most `40` characters,
- does not include spaces,
- does not include commas,
- does not include semicolons,
- does not include `=`, or
- does not include `&`;
- each character is either an alphanumeric or contained in the following string:

  `!@#$%^*()-_+[]{}:'|`~<.>/?`

If the specified username is problematic one of two things will happen:

- If the specified username is problematic because it violates one of the first 7 bullet points, the user should be told in a useful manner through the console with the specific details. It shouldn't repeat the specifics in the message more than once. All issues should be listed. One example: if `, ,=` is the value assigned to username, the console log (as one console log) should say:

```
Username must be 5 characters or longer.
Username cannot contain spaces.
Username cannot contain commas.
Username cannot contain =.
```

- If the specified username is problematic because it violates **only** the last bullet point (and none of the first 7), the user should be told with the following message:

```
Username can only use characters from the following string:
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
!@#$%^*()-_+[]{}:'|`~<.>/?
```

  This message will not show if one or more of the first seven violations are triggered.

If the specified username is acceptable, the user should be told with the following message (with the username inserted):

```
The username Sarah is acceptable.
```

Grading

Here's how your work will be graded...

- For each question, the following offenses will lead to a 1-point deduction:
  - Omitting semi-colons.
  - Failing to declare a variable using either `'let'` or `'const'`.
  - Using == or != (loose equality or inequality).

  Example: if you omit semi-colons in each question, you will score at most 17/20.

1. (6 pts) The grader will check that the calculations are correct for 3 test cases.
2. (6 pts) The grader will check 6 test cases. They will deduct at most 6 points.

   Make sure to consider the existence of `"Nusername"` and `"usernameN"`.

   Make sure that you return the empty string when you mean to. The empty string does not contain any characters.
3. (8 pts) The grader will check 8 test cases. 4 of them will be the same username from cookies in from Pb. 2.

Important: Delete all your test cases so that executing `hw2.js` does not cause anything to be printed to the console. Up to 2 points will be deducted.