1. A program contains the following line of code:

```
const x = array.filter(Boolean)
```

Explain what that line of code may be doing. *2-points*

Answer) This code presented in the question uses a method called filter which creates a new array of elements that returns true. The purpose of this code is to remove the falsy values from the array such as null and undefined

2. When working on Project 1, a student decided to experiment with JavaScript Map objects.

```
> m = new Map();
Map {}
> m.set('a', 1) //set value
Map { 'a' => 1 }
> m.get('a') //get it
1
> m['b'] = 2 //try using more convenient [] notation
2
> m['b'] //it works!
2
```

The student first tried the **`get()`** and **`set()`** methods as per the documentation, but then found that the more convenient **`[]`**-indexing operator also worked. So the student decided to do their project using the **`[]`**-indexing operator with **`Map`**'s. The project worked perfectly. It turns out that the student was wrong in using the **`[]`**-indexing operator with **`Map`**'s, so why did the project still work? *5-points*

Answer) The reason that the student was wrong in using the [] indexing operation with Maps is because in Javascript, the map function allows keys to any data types. Furthermore, the objects in Javascript inherit properties and methods through linked chains, where missing properties are searched in parent references. Therefore, since map[key] attempts to access properties on the Map object itself rather than retrieving stored values, it does not work as expected. To correctly access values in a Map, the .get(key) method must be used instead. However, the reason that the project was still able to work was because the student was adding a plain object property. Therefore, since they consistently access the values using m.get(key), everything was able to still work correctly.

Give regex's which precisely describe:

- **a. All binary strings of 4-or-more 0's?**
  - i. */[01]*0[01]*0[01]*0[01]*0[01]*/*
- **b. All binary strings of odd length containing alternating 0's and 1's.**
  - i. /^(01|01*(0|1)$/
- **c. All binary strings over 0 and 1 representing numbers greater than 5 when interpreted as binary numbers.**
  - i. */^1(1[01]*|0[01]{2,})$/*
- **d. All binary strings over 0 and 1 represent numbers which are evenly divisible by 4 when interpreted as binary numbers.**
  - i. /^[01]*00$/
- **e. All non-empty binary strings of length less than or equal to 5 containing only 0's and 1's where the number of 0's is equal to the number of 1's. 10-points**
  - i. */^(?:10|01|(?:(?:00|11)(?:11|00|10|01)))$/*

Give precise but compact descriptions for the strings described by the following regexes. If possible, try to relate the matching strings to the syntax of common programming languages.

a. /^[-+]\d+/m

i. This line of regex is for **signed integer literals** and looks for a + or - sign and then checks if there is a digit after the sign given. This is similar to the integer literals often used in languages like **C Language**

b. /0[bB][01_]+/

i. This line of regex matches for **binary integer literals.** It does this by looking if it starts with a capital or lowercase b with a 0 in the front. After, it checks if there are 0s and 1s after it. Basically, it looks if the string starts with 0b or 0B and then followed with a binary digit, either 0 or 1. This is similar to **C++**

c. /[-+]?(?:\d*\.\d+|\d+\.\d*)$/

i. This line of regex is for **both signed and unsigned floating point numbers.** This does this by checking first if it starts with a + or - and makes sure if there is a decimal point following. Finally, it checks if there is a number whether it is in front and after the decimal point. This resembles **C Language** representation of floating point numbers This is very similar to what we learned in CS220

d. /\'([^\\\'\n]|\\.)\'/

i. This line of regex makes sure that there is **only one character in between quotation marks**. There cannot be more than one or less than 1 character so empty space would not work either. It checks if the string starts with a quotation mark and then follows with a single character and a final quotation mark after. This is similar to **C Language**

e. /\"(?:[^\\\"\n]|\\.)*\"/

i. This line of regex makes sure that the string is opened and enclosed with a " mark. Thus, something like "this is a test" would work and something "this is a test would not work. This is similar to **C Language** and how we are supposed to system.out messages.

5. Here is an example of a simple HTML document:

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <title>Sample</title>
  </head>
  <BODY>
    <h1>A Sample Document</h1>
      <p class="sample-para">
        Some <strong>strong and <em>emphasized</em> text</strong>
      </p>
  </BODY>
</html>
```

The HTML contains markup within **<** and **>** angle-brackets tags. Each tag has a case-insensitive element name; ending tags start with **</**. The actual contents of the document is the content of the **body** element.

Find bugs and inadequacies in the following function which purports to extract the plain-text content of an HTML document. Then provide a fixed version of the function.

```
/** Given a string html for a HTML document, return the text content
 *  with all the HTML markup removed.  Specifically, remove header
 *  info up to and including the initial <body> tag and the footer
 *  including </body> and beyond.  Also strip out all remaining
 *  HTML tags as well as empty lines.
 */
function htmlToText(html) {
  return html.
    replace(/(.|\n)*\<body.+\>/, ''). //remove up till body
    replace(/\<\/body(.|\n)*/, ''). //remove from </body
    replace(/\<.+\>/, ''). //remove tags
    replace(/^\s*$/, '');  //remove empty lines
}
```

The extracted text for the sample document should be something like:

```
  A Sample Document
    Some strong and emphasized text
```

Answer) A bug that corresponds with the example of the simple HTML document and the given code is that the code tries to remove everything until we reach the body from the first line of code. However, relevant information can be removed from this if we have multiple body's which can cause a lot of trouble. This code is also incorrectly removing the HTML tag and is improperly removing blank lines.

Revised Code

```
function htmlToText(html) {
   let text;
   if (/<body[^>]*>/i.test(html)) {
      text = html.replace(/[\s\S]*?<body[^>]*>/i, '');
      text = (text.match(/[\s\S]*?(?=<\/body>)/i) || [text])[0];
   } else {
      text = html;
   }
   text = text.replace(/<[^>]+>/g, '');
   text = (text.match(/^(?!\s*$).+/gm) || []).join('\n');
   text = text.split('\n').map(line => line.trim().replace(/\s+/g, ' ')).join('\n');
   return text.trim();
}
```