CSC 309H1 F 2018 Midterm Test
Duration — 50 minutes
Aids allowed: none

Student Number: _ _ _ _ _ _ _ _ _ _ _ _

UTORid: _ _ _ _ _ _ _ _

Last Name: _____ **First Name:** _____

Lecture Sections: L0101 (MW 3), L0201 (MW 4)
Instructor: Mark Kazakevich

---

*Do **not** turn this page until you have received the signal to start.*
(Please fill out the identification section above, **write your name on the back of the test**, and read the instructions below.)
Good Luck!

This midterm is double-sided, and consists of 5 questions and a list of some JavaScript functions/methods. *When you receive the signal to start, please make sure that your copy is complete.*

- Assume `'use strict';` for all JavaScript.
- Comments are not required except where indicated, although they may help us mark your answers.
- No error checking is required: assume all user input and all argument values are valid, unless otherwise specified.
- If you use any space for rough work, indicate clearly what you want marked.
- **Do not remove any pages from the test booklet.**

1: _____/ 6

2: _____/ 4

3: _____/ 6

4: _____/ 4

5: _____/ 7

TOTAL: _____/27

Total Pages = 8

Question 1. [6 marks]

Indicate whether each statement is True or False by circling the appropriate answer.

TRUE FALSE The Internet Protocol does not require a pre-arranged client-server connection.

TRUE FALSE `var` has lexical scope, but `let` does not.

TRUE FALSE Setting the prototype of object `a` to be object `b` does not create a copy of object `b`.

TRUE FALSE Absolute positioning is good for positioning some text inside a statically positioned `<div>` element.

TRUE FALSE Reliable protocols do not react to packet loss.

TRUE FALSE Anonymous JavaScript functions can only be executed as callbacks.

Question 2. [4 marks]

The following two lines of JavaScript are run, creating two variables in the global scope:

```
let a = 4;
let b = 1;
```

The code fragments below are each run directly after the above two lines. They are run **independently** of each other. Beside each code fragment in the table below, write the console output when the code fragment is executed after the above two lines. If the code would cause an error, write ERROR and give a brief explanation.

Code	Output or Cause of Error

`(function () {`
`  if (a > 2) {`
`  let b = 3; a = 0;`
`  }`
`  console.log(a + b);`
`})();`	

`const c = 0;`
`for (let i = 0; i < a; i++) {`
`  if (i > 2) { b = 0; }`
`  c = c + b;`
`  console.log(c);`
`}`	

`function foo() {`
`  function bar() {`
`  return function () { console.log(a); }`
`  }`
`  a = 3;`
`  return bar();`
`}`
`const baz = foo(); baz();`	

`function f1() {`
`  const n = b;`
`  return function (p) { return n + p; }`
`}`
`const f2 = f1();`
`console.log(f2(a));`	

Question 3. [6 marks]

The following are the contents of the file `accounts.js`. Under it is a set of commands executed in the JavaScript console after the file has been run. Fill in all of the boxes in the file and the console output such that the console commands all work correctly and show the correct output.

```
function Account() {
   this.balance = 0;
}

[                                                  ] = {

    buyWithCard: function(amount) {
        if (this.balance >= amount) { this.balance -= amount; console.log('All good.'); }
        else { console.log('Not enough money!'); }
    },
    [                                                                                  ]

}
----end of accounts.js----------------------------------------------------------------------
----JavaScript Console below----
<accounts.js is run>
> const myAcc = new Account();
> myAcc.showBalance();
0
> myAcc.addToBalance(30);
> myAcc.showBalance();
30
> myAcc.buyWithCard(20);
[                                        ]
> myAcc.showBalance();
10
> myAcc.buyWithCard(50);
[                                        ]
> const yourAcc = new Account(); myAcc.buyWithCard.bind(yourAcc)(5);
[                                        ]
```

Question 4. [4 marks]

Consider the JavaScript code below.

```
function foo0() { console.log(0); }
function foo5() { console.log(5); }

setTimeout(foo5, 5000);  // 5000 milliseconds = 5 seconds
setTimeout(foo0, 0);
console.log('bar');
```

Part (a) [1 mark]

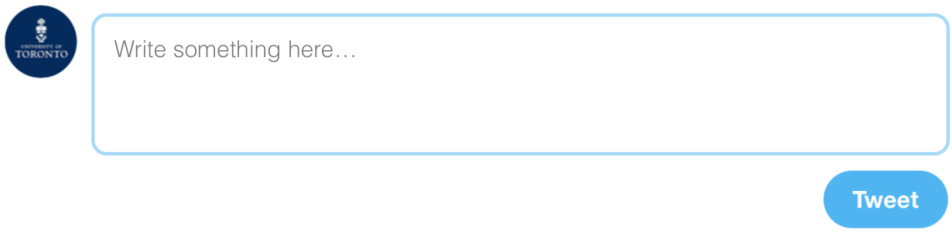
Write the console output from executing this code (output order matters). Assume enough time has passed for everything to print.

Part (b) [3 marks]

Suppose we are running this code in the browser. Using the JavaScript event loop, explain the order of the console output of this code. (You don't have to use the exact terminology we used in class for every part of the event loop, but your explanation should clearly explain what is going on).

Question 5. [7 marks]

Below is a simple form layout for posting a Tweet to a timeline.



The HTML for this form section is displayed below (assume all CSS has been properly written and loaded):

```
<div id='tweetForm'>
    <img id='pic' src='profilepic.png'>
    <input id='tweetInput' type='text' placeholder='Write something here...'>
    <button class='tweetButton'>Tweet</button>
</div>
```

Tweets are added to the **bottom** of a tweet timeline only when clicking the 'Tweet' button. Below is the HTML for the tweet timeline, as well as one tweet as an example. The text the user wrote in the box in the form is the `'Tweet text here.'` in the example tweet. The next tweet to be added would go under this tweet as its sibling element.

```
<div id='timeline'>
    <div class='tweet'>
        <img src='profilepic.png'>
        Tweet text here.
    </div>
</div>
```

If the 'Tweet' button is clicked and the text is greater than 280 characters long, no tweet is added to the timeline, and instead the following element is added as the last child of the tweet form (the one with id `tweetForm`).

```
<div id='errorMessage'>
    <span>Max 280 characters.</span>
</div>
```

On the **next page** is a JavaScript file, `tweets.js`. Fill in the boxes to create the appropriate elements as shown above, and to ensure that the described user interactions will work as intended.

All DOM elements and text nodes must be created dynamically. You may not use .innerHTML, .innerText, or anything similar with hardcoded text. Doing so will result in a 0 for this part.

The back of this test has a list of some DOM manipulation methods for reference.

```
tweets.js
-------------------------------------------------------------
const form = document.querySelector('#tweetForm');
const timeline = document.querySelector('#timeline');
form.                                  ('click',                          );
function addTweet(e) {
    if (e.target.classList.contains(                        )) {

        const textInput =                                          

        if (textInput.value.length <= 280) {
            const tweet = makeTweet(textInput.value);
            timeline.appendChild(tweet);
        } else {
            const error = makeErrorMessage();
                                                 
        }
    }
}
function makeTweet(tweetText) {
    const tweet = document.createElement('div');
                                                                                
                                                                                
                                                                                
                                                                                
                                                                                
                                                                                
                                                                                
    return tweet;
}
function makeErrorMessage() {
    const error = document.createElement('div');
                                                                                
                                                                                
                                                                                
                                                                                
                                                                                
                                                                                
                                                                                
    return error;
}
```

Last Name: _____ **First Name:** _____

JavaScript DOM functions/methods

```
document.querySelector(selector)
document.querySelectorAll(selector)

document.createElement(string)
document.createTextNode(text)
element.appendChild(element)

element.setAttribute(attributeName, value)

element.addEventListener(event, function)
event.preventDefault()

Properties:
element.className
element.id
element.classList
element.value


array.push(object)
```