# EECS 1012: LAB 09 – jQuery basics. 2D arrays. Learning kits

## A. REMINDERS

- This lab is due on Aug 9 (Tuesday) at 11pm.
- The pre-lab mini quiz is considered part of this lab.
- You are required to complete the pre-lab mini quiz 9 posted on eClass. This pre-lab is due at the same time as the lab (Aug 9th, 11pm)
- You are welcome to attend the lab session on Aug 5, if you stuck on any of the steps below. TAs and instructor will be available to help you. The location is WSC105. Attendance is optional. You can come to either the morning or the afternoon lab sessions.
- Feel free to signal a TA for help if you stuck on any of the steps below. Yet, note that TAs would need to help other students too.
- You can submit your lab work any time before the specified deadline.

## B. IMPORTANT PRE-LAB WORKS YOU NEED TO DO BEFORE GOING TO THE LAB

1) Download this lab files and read through them carefully.
2) You should have the basic understanding of jQuery
    - Events (such as `click`) and event handlers
    - `$(#id).html(),    $(#id).css(style, value)  $(#id).attr(attribute, value)`
    - Search w3school jQuery tutorials and the like, to investigate how to use the jQuery functions.
3) You should have basic understanding of 2D arrays, which can be considered as array of arrays. Understand how to access the element of a 2D array, how to get number of rows and number of columns in the 2D array.

## C. GOALS/OUTCOMES FOR LAB

- To practice operating on 2D arrays.
- To practice using jQuery.

## D. TASKS

1) Your first task in this lab is to implement several simple JS functions to change the text and css style of html components.
2) Your second task is to implement the same html page using jQuery.
3) Your third task is to develop an flowchart and JS implementation for 2D arrays.
4) Your third task is to generate checkerboxes, which internally us 2D arrays.
5) Submit your learning kit with at least 30 questions.

## E. SUBMISSIONS

eClass submission. More information can be found at the end of this document.

## F. COMPUTATIONAL THINKIG

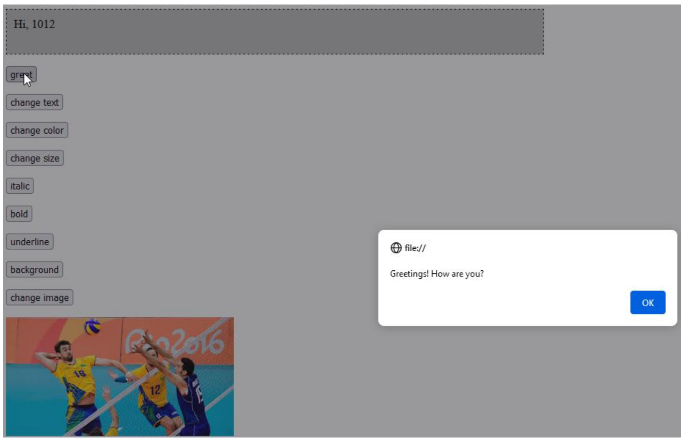
### Task 1: simple JS (using material that we covered in the first half of the term).

You are given html file **lab9.html** and **lab9.js**.  you don't need to modify the html file.

Implement the various JS functions following the comments there, using the JS knowledge we learned in the first half of the term.

The effects of the functions are shown in the following screenshots. Note that you can click the buttons in different orders. Illustrated here are the expected results when the buttons are clicked in order.

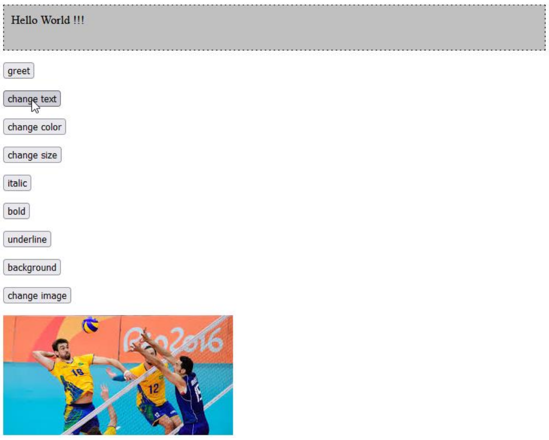
Also note that this is a simple implementation so if you click the same button multiple times, it will not make changes any more. Refresh your browser to start over.



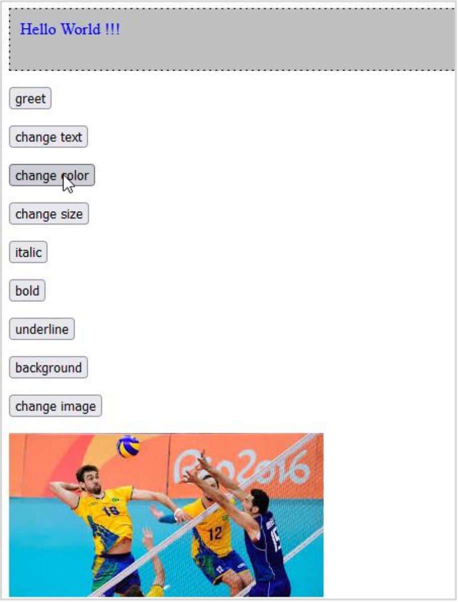
Hi, 1012

file://

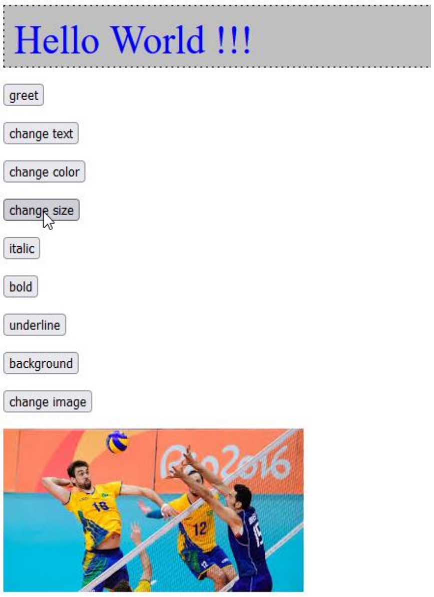
Greetings! How are you?



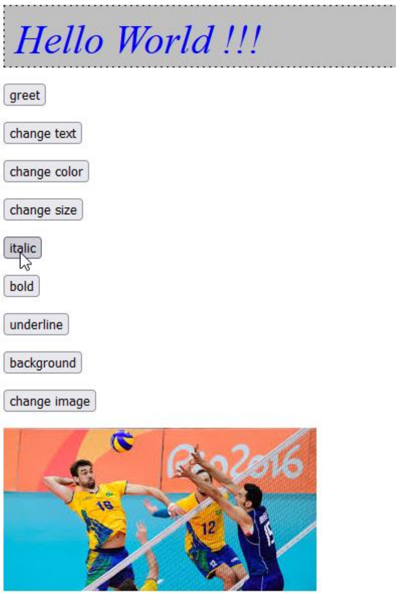
Hello World !!!



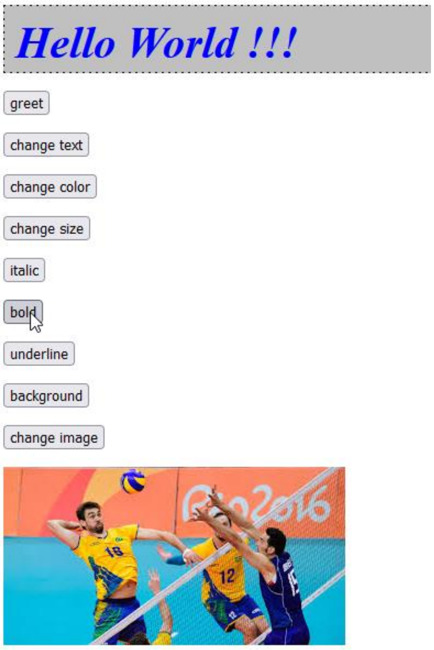
Hello World !!!



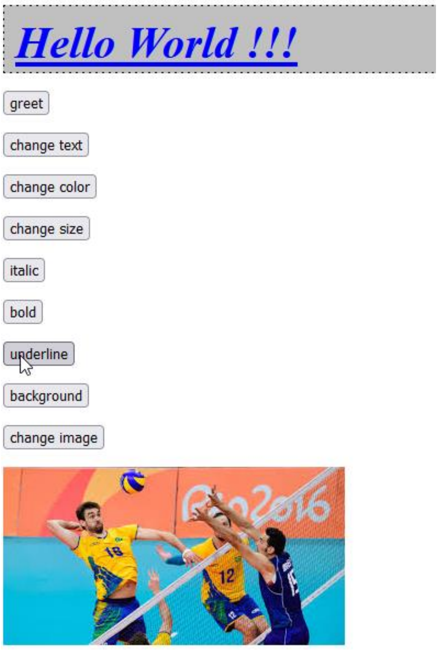
Hello World !!!



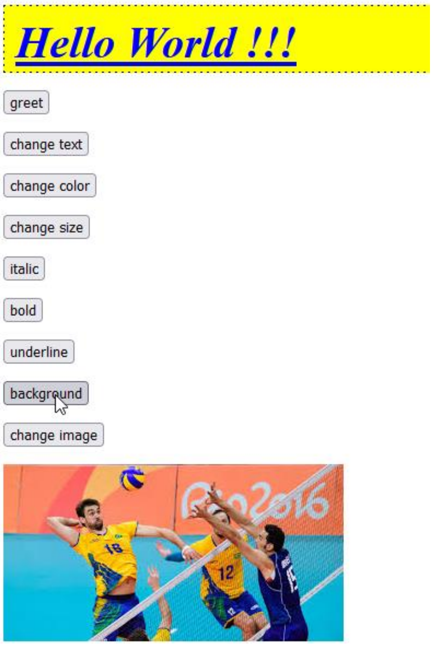
*Hello World !!!*



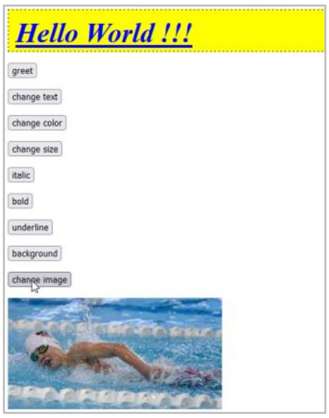
***Hello World !!!***



***Hello World !!!***



***Hello World !!!***



Hello World !!!

**Task 2: Simple jQuery**

Accomplish the same functionality using jQuery.

You are given **lab9jQ.html lab9jQ.js.**

In html, add line <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script> in the head section of the html file. This allows the program to use jQuery library online. **Note that this line should be added before the link to the JS file**, so that the jQuery codes can be recognized.

Add buttons to html as in Task1. Note that you should not add onclick event on any buttons. We will use jQuery to do this.

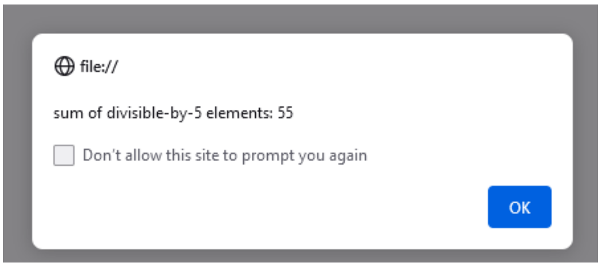
In **lab9jQ.js**, implement the function so that when the buttons are clicked, they do the same job as in task1. Use the jQuery syntax introduced in class. E.g., should not use getElementByID("…"). or innerHTML=…

Hint: in changing image src, use **attr()** function of jQuery.

**Task 3: 2D array**

First, develop a flowchart for an algorithm to receive a 2D array of integer numbers, and output the sum of elements in the array that are divisible by 5. Save the flowchart as an image.

Then, open **checkerboard.html** and **checkerboard.js**. Study the code of **array2D()** and sub-function **display(arr)** and understand how to visit 2D array elements one by one. Then, implement the sub-function **sum(arr)** which takes as argument an 2D array of integer numbers, and returns the sum of elements in the array that are divisible by 5. The returned value is used in caller function **array2D()**. If implemented correctly, the program will output 55 in an alert window (5+10+15+25 =55).



file://

sum of divisible-by-5 elements: 55

☐ Don't allow this site to prompt you again

**Task 4: 2D array, jQuery**

You are given html, css and js files for drawing checkerboard. You don't need to modify html and css files.

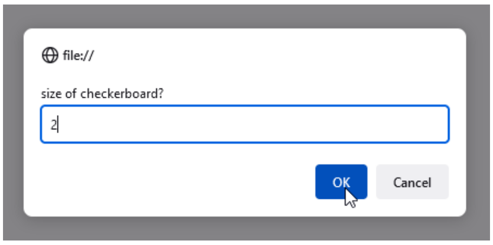
Complete the function **drawTheCells()** in **checkerboard.js**, which promotes user for a size n of matrix, and then draws a n by n matrix with the top-left square in red, and the alternating color of red and black, as shown in th the following figures.

In your solution, you are given a sub-algorithm **drawOneCell(color)**, which uses jQuery to add buttons to the div section with the specified color. You should call this function as **drawOneCell("red")** or **drawOneCell("black")**.

Hint1: you will need nested loops in your solution to set cell by cell.

Hint2: for each cell color, think about the pattern when a red color appear: in even rows (0,2,4..), in what columns does red appear; In odd rows, in what columns does red appear….

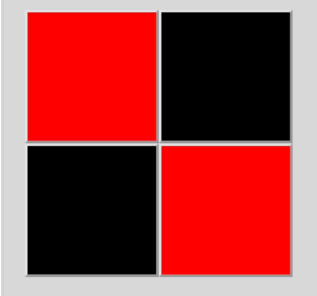
Hint3: after displaying a row, call **drawNewlLine**() so that the next row is displayed on the next line.



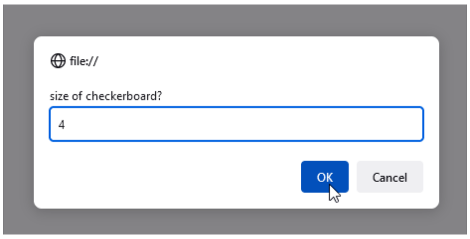
file://

size of checkerboard?

2



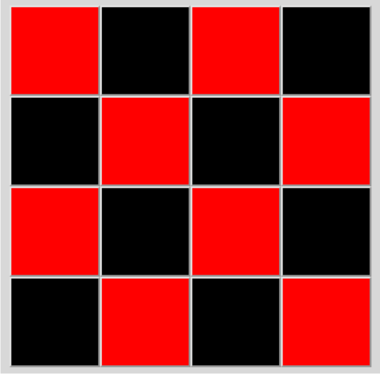
2 by 2 matrix



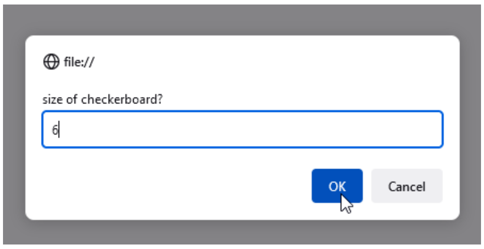
file://

size of checkerboard?

4



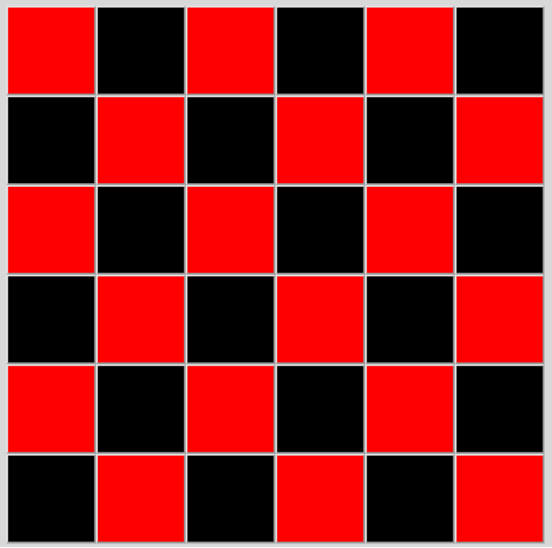
4 by 4 matrix



file://

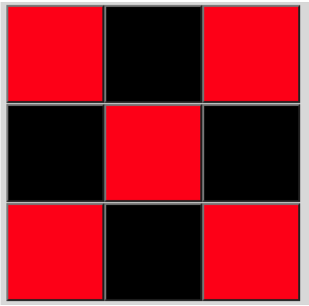
size of checkerboard?

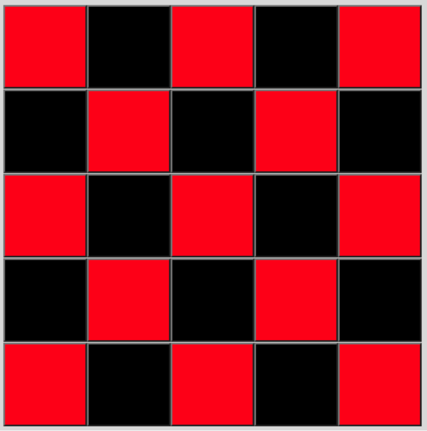
6

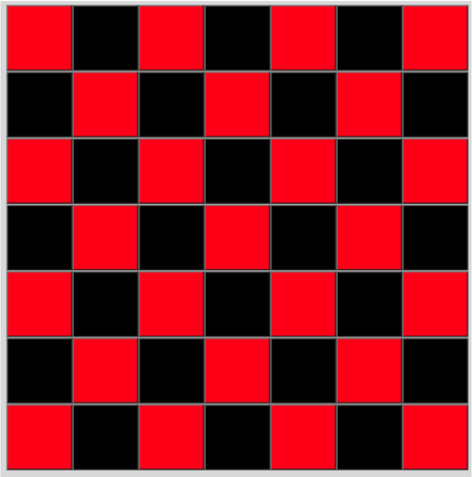


6 by 6 matrix

**Matrix of sizes 3 by 3, 5 by 5 and 7 by 7 are also shown:**







**Task 5: Complete your learning kits with at least 30 questions.**

- As discussed in class, you should have at least 30 buttons, one for each different question. On clicking the button for a question, the description of the problem is shown in the textbox. Then when clicking "Design" checkbox, the flowchart for this question is shown. When clicking the "JavaScript solution" checkbox, the JS code is shown.
- When clicking another button, as in lab3, current texts, images should all be cleared.
- You can add any algorithms, not necessarily lab questions.
- No need to do "Another Solutions" (welcome to do)

- The problems can be from different resources
  - could use any previous problems you have done: from labs, assignments, slides, tests, labtests, classes..
  - could also use exercise questions given in 'AFTRER-LAB WORKS'
  - could also use exercise questions mentioned in slides
  - could tweak the above questions to create new questions
  - could come up with own questions, or search online for some questions
  - ....
- Should contain problems of various types:
  - should have problems that have a *loop* without nesting.
  - should have problems that have a *nested loop*.
  - should have problems that involve *arrays*.
  - should have problem that call other sub-*functions*.
  - should have problems that are recursive.
- Welcome to have problems that involve *2D arrays*.
- should not use trace table to replace flowchart, except for recursive algorithms. For non-recursive algorithms, must give flowchart.
- don't need to do run button (welcome to do)

An example which we showed before, is shown in the next page.

## G. AFTER-LAB WORKS (THIS PART WILL NOT BE GRADED)

1) Given a 2D array, draw flowcharts to find max value in the array. Implement in JS code.
2) Given a 2D array, draw flowcharts to display (or count the number of, or calculate average of) prime values in the array. Implement in JS code.
3) Given a 2D array, draw flowcharts to display (or count the number of, or calculate average of) odd values in the array. Implement in JS code.
4) Given a 2D array, draw flowcharts to display (or count the number of, or calculate average of) values in the array that are divisible by 7. Implement in JS code.
5) Given a 2D array, draw flowcharts to display (or count the number of, or calculate average of) values at even rows. Implement in JS code.
6) Look at more 2D array examples posted on eClass.

## H. Submissions --- two zip files

- You should already have a **Lab09** folder that contains the following files:
    - **lab9.html, lab9.js, lab9jQ.html, lab9jQ.js** and the two images.
    - **An image file for the flowchart**
    - **checkerboard.html, checkerboard.js, checkerboard.css**

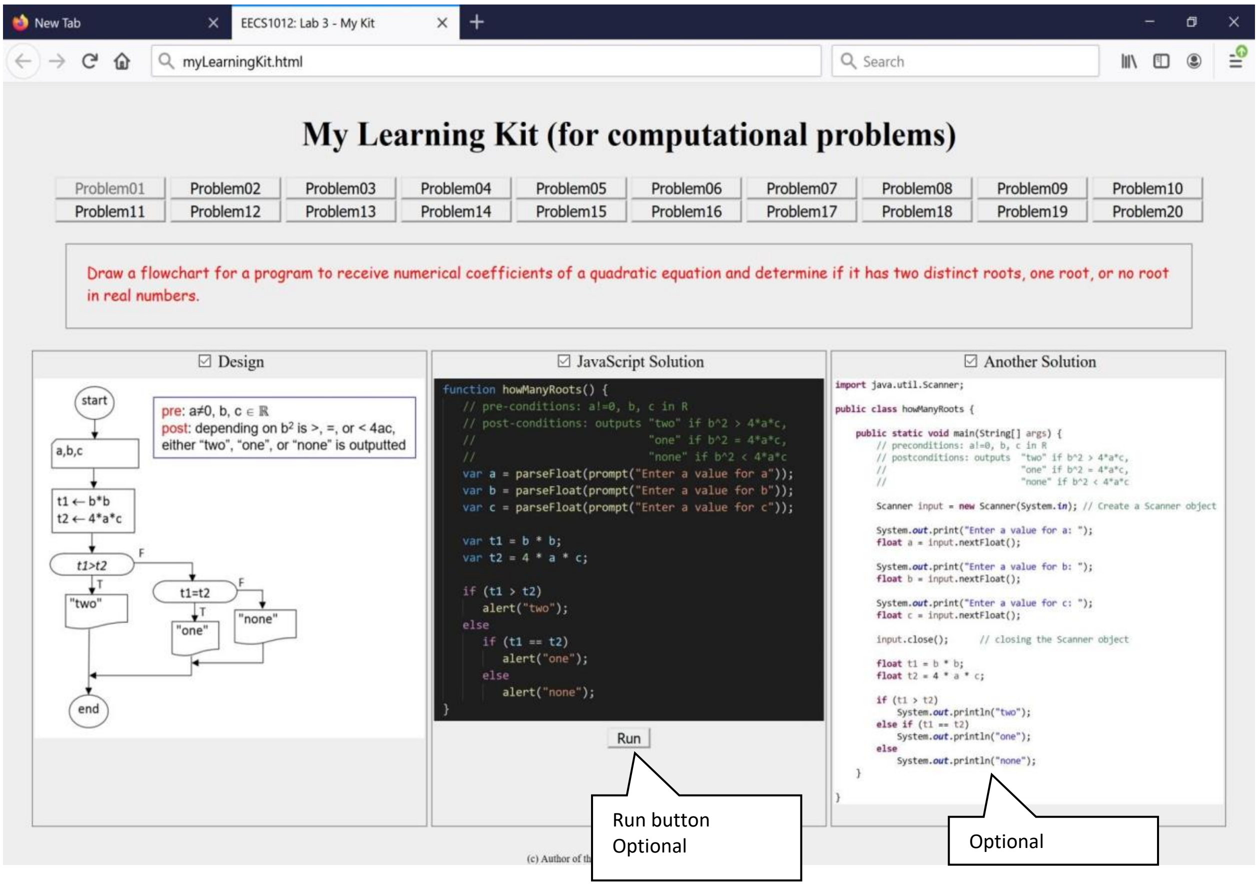
Your HTML should pass the HTML validator at https://validator.w3.org

**Compress the Lab09 folder (**.zip **or** .tar **or** .gz**), and then submit the compressed file to the lab9 submission page on eClass.**

- **Compress your learning kits files (**zip **or** tar **or** .gz**) and also submit to the lab9 submission page on eClass.**

---

An example of learning kit. You can have other style.



My Learning Kit (for computational problems)

Draw a flowchart for a program to receive numerical coefficients of a quadratic equation and determine if it has two distinct roots, one root, or no root in real numbers.

☑ Design

pre: a≠0, b, c ∈ ℝ
post: depending on $b^2$ is >, =, or < 4ac, either “two”, “one”, or “none” is outputted

☑ JavaScript Solution

```
function howManyRoots() {
   // pre-conditions: a!=0, b, c in R
   // post-conditions: outputs "two" if b^2 > 4*a*c,
   //                          "one" if b^2 = 4*a*c,
   //                          "none" if b^2 < 4*a*c
   var a = parseFloat(prompt("Enter a value for a"));
   var b = parseFloat(prompt("Enter a value for b"));
   var c = parseFloat(prompt("Enter a value for c"));

   var t1 = b * b;
   var t2 = 4 * a * c;

   if (t1 > t2)
      alert("two");
   else
      if (t1 == t2)
         alert("one");
      else
         alert("none");
}
```

☑ Another Solution

```
import java.util.Scanner;

public class howManyRoots {

    public static void main(String[] args) {
        // preconditions: a!=0, b, c in R
        // postconditions: outputs  "two" if b^2 > 4*a*c,
        //                          "one" if b^2 = 4*a*c,
        //                          "none" if b^2 < 4*a*c

        Scanner input = new Scanner(System.in); // Create a Scanner object

        System.out.print("Enter a value for a: ");
        float a = input.nextFloat();

        System.out.print("Enter a value for b: ");
        float b = input.nextFloat();

        System.out.print("Enter a value for c: ");
        float c = input.nextFloat();

        input.close();      // closing the Scanner object

        float t1 = b * b;
        float t2 = 4 * a * c;

        if (t1 > t2)
            System.out.println("two");
        else if (t1 == t2)
            System.out.println("one");
        else
            System.out.println("none");
    }

}
```

(c) Author of th

Run button
Optional

Optional