Name: _____

ID: _____

Midterm

CS193C, Summer 2017, Young

As we've discussed, you have three hours to complete this exam and get it submitted. Please get it online on Canvas by 9:30pm. Use the same submission procedure as for your homework assignments. Make sure your files are up at 9:30pm. If your files are more than a few minutes late, you can expect to get a zero. Remote SCPD students and OAE students submit to "Midterm Makeup" on Canvas, regular students submit to "Midterm" assignment on Canvas.

This test is open book, open note. You may use the Internet to access reference material, but may **not** communicate with anyone in any way (electronic or otherwise) other than the CS193C teaching staff. **Do not post questions or discuss the midterm on Piazza** either during or after the midterm. We have remote SCPD students taking the exam later this week, and we do not want comments on the midterm visible to them before they take the exam.

The Stanford Honor Code

1. The Honor Code is an undertaking of the students, individually and collectively:

    a. that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;

    b. that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

2. The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

3. While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work

I accept the letter and the spirit of the Stanford University Honor Code. I swear that I have not given or received unpermitted aid on this exam, and I have taken an active part in stopping any violations of the Honor Code which I see on this exam.

Signed: _____

Instructions

- **Make absolutely sure you turn everything in on time. Do not turn in your exam late.**

This exam is due at **9:30pm**.

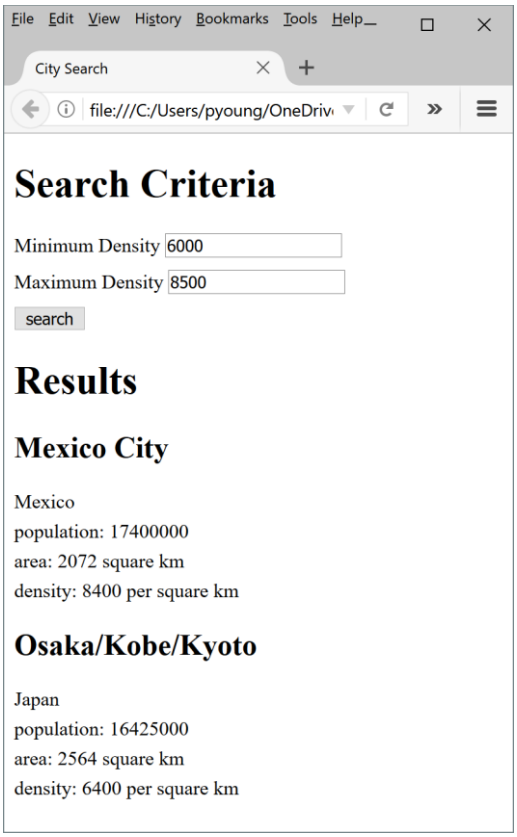
- We will be testing your exam solutions in either Firefox or Chrome, **your choice**. If you want us to test in a specific browser add a paragraph at the top of each webpage telling us which browser you want us to use.
- **We are interested in running code only.** If your code doesn't work, you should expect to get no points. Whatever you turn in needs to run. It doesn't need to run perfectly, so if the numbers generated aren't quite correct, or if things don't move the way the problem asks, you can certainly get partial credit. But we aren't giving partial credit for unfinished code that doesn't actually do anything in the web browser.
- **You may not use jQuery or any other library code.** If you are found using jQuery or other libraries on a problem, you will not receive credit on the problem.
- If we do not provide values for items you need, any reasonable value is acceptable. For example you may make up your own alt values for images or set widths and heights as desired when they are not specified in the problem statement.
- If desired you may use the convenience innerHTML property along with any other techniques which are supported by whichever web browser you've asked us to test on, even if they are not part of the official W3C standard.
- Make sure everything validates.
- You may not use Dreamweaver or other WYSIWYG editors.
- All code must be developed from scratch – this means you may not copy directly from code you find on the Internet. You may use CS193C lecture examples I have uploaded to Canvas directly.
- If in doubt, cite what you've done by explicitly putting a paragraph visible on your webpage and it won't be considered an honor code violation, although you may lose points. You can put this directly below the "grade in Chrome" or "grade in Firefox" paragraph.
- As noted in Handout 2 "Computer Science and the Stanford Honor Code" we reserve the right to use automatic plagiarism detection, comparing your solution to other students' solutions and to code on the Internet (the Judicial Affairs office has ruled that this is permissible).
- All problems are worth the same amount of points. Although some are definitely harder than others. Plan accordingly.
- *I'm not expecting most students to complete all the questions. So don't panic if you don't think you're going to finish the exam. This is expected.*

- **Keep a working backup copy of your code at all times. When you have successfully implemented a feature, create a backup copy of your files. You do not want to be rushing to add a new feature to a problem when time runs out and discover that you've broken your previously working webpage and don't know how to get it working again.**

- **Make sure you've double and triple checked your submission zip file before submitting it, it should contain all files needed to run your programs, including any files we've provided.**

City Search

For this problem we search a list of international cities and display the cities that match a particular search criteria. We will be searching for cities based on their population densities. Here's a screenshot showing the webpage in action:



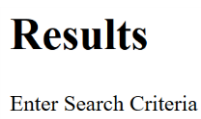
The list of cities and their corresponding data is contained within the "cities-list.js" file which is part of the midterm downloads. You must load this file into your HTML file using a <script> tag, you may not change the format the data is stored in, and should not modify this file. If you want to write your JavaScript code in an external JavaScript file, please use a second JS file.

The title of this webpage is "City Search". Save the html file using the name "cities.html". Names for any CSS or JavaScript files in addition to "cities-list.js" you use (if any) are up to you, just make sure "cities.html" properly loads them.

As you can see the webpage contains some text fields allowing the user to specify a minimum and maximum density of cities to display. When the user clicks on the "search" button display all cities that match the given criteria. We don't care about the order with which you displayed the cities as long as the correct cities are ultimately displayed. If either text field is blank, that means that the user does not care about that particular criteria. If both text fields are blank, display all cities in the city list when the user clicks on the "search" button. If there are no cities which meet the criteria, show the search results as "No Cities Found".

You may assume that the user either leaves the text fields completely empty or that they enter zero or positive integers into them. You may assume that city populations are numbers somewhere between 0 and the maximum positive number which may be stored in JavaScript. A city population may legally be 0 (see for example California Ghost Towns).

When the webpage loads, both text fields should be blank, and the search results section should display: “Enter Search Criteria” like this:



You do not need to have a pixel perfect representation of the screenshot shown above. However, the titles “Search Criteria” and “Results” should be the largest items on the webpage, city names should be somewhat smaller than “Search Criteria” and “Results”, but larger than the rest of the text. Make sure it is clear to the grading TA what each text field is for. Show all city data in the order shown on the screenshot – city name, country name, population, area, and density, each on their own line, and include the “square kms” and “per square kms” units for area and density as shown in the screenshot.

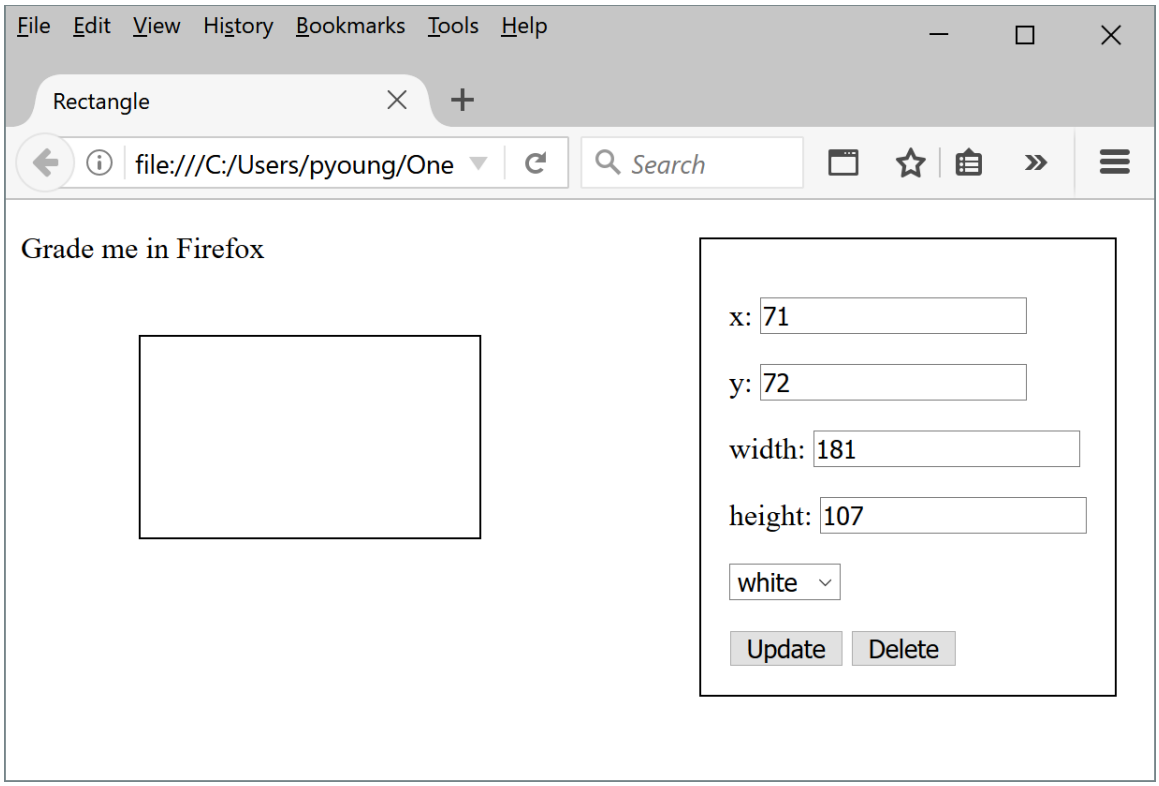
Don’t worry about your margins, or the spacing between items.

Rectangle

For this problem you will use JavaScript to allow the user to draw a rectangle on the webpage. The user can initially place the rectangle on the page by clicking twice. The first click defines one corner of the rectangle, and the second click defines the diagonally opposite corner of the rectangle. Initially the rectangle will appear as a white rectangle with a 1-pixel black border. However, controls provided allow the user to manipulate the rectangle, change its interior color, as well as allowing them to delete the rectangle.

The title of this webpage is "Rectangle". Save the html file using the name "rectangle.html". CSS and JavaScript file names (if any) are up to you, just make sure "rectangle.html" properly loads them.

Here's a screenshot of the webpage in action. This screenshot shows the page after the use has begun interacting with it. When the webpage initially loads, no rectangle will be displayed until the user clicks twice.



As you can see the control section includes text fields allowing the user to enter x, y, height, and width. As soon as the user creates a new rectangle, these data items should be filed in by your program with the data defined by the user's clicks. Regardless of which corners of the rectangle the user clicked to define the rectangle, x, y always corresponds to the top-left corner of the rectangle. The control section also includes a pull down menu of colors which can be used to change the interior color of the rectangle. The color menu includes the colors white, red, green, and blue. When a user creates a new rectangle, this should be set to display the white option. The rectangle always has a 1-pixel solid black border regardless of the interior color selected from the menu.

The control section includes two buttons – "update" and "delete". Changes to the text fields or color menu are ignored until the user clicks on the "update" button. Once clicked on, the rectangle will move, resize, and change color to match the current entries. If the rectangle isn't currently present (either because the webpage has just loaded or because it was deleted), clicking on update will have no effect.

Clicking on the "delete" button will completely remove the rectangle. When a rectangle has been deleted, you can leave the text fields filed with the previous data (or if this offends your sensibilities, feel free to empty or zero out the text fields when the rectangle is deleted).

You may assume that the user only enters legal numbers in the x, y, height, and width text fields. Height and width will always be positive integers (they may not be zero). The x, y coordinates will always be integers (but may be zero or negative). These coordinates will be relative to the top-left corner of the window. The user may move the rectangle either partially or completely outside of the viewing area of the window. They may also resize the rectangle so that it is partly outside the viewing area of the window. We mostly don't care what the behavior of the program is when the rectangle overlaps with the control area, the rectangle can display above or below the controls and it is okay if it interferes with the behavior of the controls. Just make sure the program doesn't crash or display an error message in this situation. It's okay if overlapping the rectangle and the controls requires a reload of the webpage because the controls are completely obscured.

We don't care how you create the labels and controls within the control section. Just make sure we can tell which text fields correspond to which purposes. In addition, surround the control section with a 1-pixel black border to clearly delineate it from the rest of the webpage. (I also added 15 pixels of padding to make it look nicer for the screenshot, but that isn't necessary.)

Once the rectangle has been placed on the webpage, mouse-clicks (other than those involving interaction with the controls in the control section) will be ignored until the user deletes the rectangle. Once deleted, mouse clicks can again be used to create a new rectangle on the webpage.

You may assume that the user does not click anywhere within the control section when defining the placement and outlining the size of a new rectangle.

The control section should stay fixed with its right side 20 pixels from the right side of the window and its top 20 pixels from the top of the window. You may use either JavaScript or CSS to make sure it stays on the far right.

Note that the user can <u>not</u> move the rectangle around with the mouse. The only way to move the rectangle is by changing the x and y text fields in the control section and then clicking on the "update" button.

Implementation Comments

Hint: If you find that your program treats the click on the delete button as the first click defining a new rectangle, what's happening is that a click on the button is being passed on to the parent of the button, which interprets it as the first click in your pair of rectangle defining clicks. You do need to fix this. There are several ways to stop this, but you may find the stopPropagation method we discussed in our lecture on events useful. It's briefly documented here:

https://www.w3schools.com/jquery/event_stoppropagation.asp

If you don't remember our lecture at all and want a detailed explanation you can find one here:

https://javascript.info/bubbling-and-capturing

Shooting Gallery

For this problem we create a simple shooting gallery game. Targets will appear briefly in the target area. The user will attempt to move the mouse on top of a target and press the mouse button down. If they are successful, the target will change to indicate that it has been hit and the user will score points.

The title of this webpage is "Shooting Gallery". Save the html file using the name "shooting.html". CSS and JavaScript file names (if any) are up to you, just make sure "shooting.html" properly loads them.

Here's a screenshot taken midway through a game, the blue 10 square is a target:

Overall Layout

The two main components of this webpage will be a target area, which is 800 pixels wide by 400 pixels tall with a 1-pixel wide black border. The target area will appear 200 pixels from the left corner of the window and 200 pixels below the top of the window.

To the right of the target area will be the information section. This will display "Current Score" and "Targets Remaining" labels and numbers. Start the "Current Score" as 0 when the webpage loads. Set the "Targets Remaining" to 7 when the webpage loads. When the game is not running, this section will also display a "Play Game" button. This button should disappear when the game is running and reappear when the game has completed. (Note that the button is not shown in the screenshot, as the screenshot was taken with the game in progress.)

The font used for the information section is up to you as is the exact placement. (I used 20 point, sans-serif, bold for the screenshots, but you can just leave it with the default font).

The exact placement of the information section is up to you, but it should appear to the right of the target area (I used a 50-pixel margin between the two sections). Both the target area and information section are fixed, and will not move in response to resizing of the browser window. They should stay in the same position relative to one another even if the window is too small (a scrollbar should appear).

General Game Flow

When the webpage loads the game is stopped. The game will begin when the user clicks the "Play Game" button.

Once the user clicks on "Play Game" the "Play Game" button will disappear and a sequence of seven sequential targets will appear. Each target will appear alone, followed by the next target 5-second later, with the first target appearing 5 seconds after the "Play Game" button is pressed. Targets will only appear briefly and then will disappear. The user will try to press the mouse button down on top of each target before it disappears. If they succeed, they will earn score points based on the target type (see target details below) and their "Current Score" will be updated. Regardless of whether or not they successfully hit the target, the "Targets Remaining" will be decremented when the target disappears. When the "Targets Remaining" drops to zero, the game is over.

When the game ends, leave the "Current Score" showing however many points the user earned. Leave the "Targets Remaining" at zero. Make the "Play Game" button appear.

Targets

The game has three different target types which may appear. These targets vary in point value, in size, and in the length of time they will remain visible. When 5 seconds have passed either since the game started or since the previous target was displayed, randomly choose one of the target types, taking the appearance probabilities (shown in the table below) into account, and display that target randomly somewhere in the target area. The target should be placed so that it fits entirely within the target area (e.g., it should not appear at a coordinate that only allows part of the target to fit within the target area).

Note that the 5-second interval starts when a target appears (*not* when it is hit or when it disappears).

Here is a table showing the three target types, their probability of appearing, point values, sizes, and amount of time they will remain visible:

Probability	Point Value	Time Visible (in milliseconds)	Dimensions (in pixels)	Image File	Image File when Hit
55%	10	2000	120x120	ten.png	ten-hit.png
35%	20	1000	80x80	twenty.png	twenty-hit.png
10%	50	500	50x50	fifty.png	fifty-hit.png

Each target has a both a regular image and a target-hit image included with the midterm downloads. If the user successfully presses the mouse down on top of a target, switch the regular image to the target-hit version of the image. After a successful hit, leave the target-hit image visible for 800 milliseconds after the hit, regardless of the original time visible for the target.

Increment the score and decrement the number of targets remaining as soon as the user clicks on the target. This does lead to a situation where the user can restart the game before the last target-hit image disappears (since that won't happen until 800 milliseconds later). You can ignore this situation for this exam. (One

possible solution for this in a real game would be to only display the "Start Game" button after the final image disappeared, but you don't need to do this for our exam.)

Implementation Comments

You'll want to use the built in random number and timing functions you used for the matching homework problem. For reference here are links to their documentation:

https://www.w3schools.com/js/js_timing.asp

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random

Feel free to copy the getRandomInt method from Mozilla directly. Reproduced here:

```
function getRandomInt(min, max) {
  min = Math.ceil(min);
  max = Math.floor(max);
  return Math.floor(Math.random() * (max - min)) + min;
   //The maximum is exclusive and the minimum is inclusive
}
```

Implementation Order Recommendation

As discussed in class, many of you will not finish this problem. You should not be concerned if you do not finish, this does **not** mean that you will fail the class. However, you should try to get as many parts of the problem working as possible. This recommended order of implementation will provide you with some structure in getting as much of the problem working as possible.

There are many different ways to implement this problem. Feel free to ignore the recommendations here. Just remember you will not receive credit for non-working code. As soon as you implement a feature and are satisfied it is working properly, make a backup copy of your file. Make sure whatever you submit on Canvas works, whether it implements all features correctly or only some of the features.

This recommended order does not necessarily cover all the features, so once you've gone through it, go ahead and go back over the instructions and cleanup any missing items.

These instructions are provided "as is". We will not give hints or clarification on this recommendation section. just providing this section is giving you a very big boost.

I recommend you leave randomizing the location of the targets to near the end, instead place all targets in the top-left corner of the target area. This will make it easier to test. Initially be prepared to reload the page frequently to reinitiate testing.

- Setup the webpage so that all parts of the user interface are visible, except for the targets.
- Make a 10 point target visible in the top-left corner of the target area when the webpage loads.
- Add an event handler such that clicking on the target changes the target to the "target hit" version of the image. Update the score and number of targets remaining as appropriate for a target having been hit. Reload the webpage as needed for testing.
- Set things up so that the target disappears in 2000 milliseconds *if the target is not hit*.
- Set things up so that the target disappears entirely 800 milliseconds *after the target is hit*.
- Generalize your code to handle all three types of targets.
- Add code to choose and display a target every 5000 milliseconds.
- Randomize the location of the target.
- Change the code so that the webpage starts with no target present. Setup the "Start Game" button to begin the sequence of targets and to have the "Start Game" button disappear once it is clicked on.
- Setup the game so that once the remaining targets hits zero the targets stop appearing and the "Start Game" button reappears.