Name: _____

Quiz Section: _____ **TA:** _____

Student ID #: _____

Rules:

- You have **110 minutes** to complete this exam.
  You may receive a deduction if you keep working after the instructor calls for papers.
- This test is open-book/notes. You may use any paper resources <u>other than practice exams</u>.
- You may *not* use any computing devices, including calculators, cell phones, or music players.
- Unless otherwise indicated, your code will be graded on proper behavior/output, not on style.
- Please do not abbreviate code, such as writing ditto marks ("") or dot-dot-dot marks (...).
- If you enter the room, you must turn in an exam and will not be permitted to leave without doing so.
- You must show your **Student ID** to a TA or instructor for your submitted exam to be accepted.

Good luck! You can do it!

Problem	Description	Earned	Max
1	HTML / CSS Tracing		25
2	PHP		25
3	JavaScript / DOM		25
4	SQL		25
TOTAL	**Total Points**		**100**

1. HTML / CSS Tracing

Draw a picture of how the following HTML/CSS code will look when the browser renders it on-screen. Assume that the HTML is wrapped in a valid full page with a `head` and `body`. Indicate a non-white background by shading lightly or by drawing diagonal lines like this. It is possible that some CSS rules shown will not apply to any elements.

HTML:

```
<p> hello </p>
<div>
    <div> one </div>
    <div id="div"> two <br /> two two </div>
</div>
<div class="div"> three <br /> three three <br /> three three three </div>
<div class="div"> four <br /> four four <br /> four four four <br />
                <span class="div"> four four four four </span> </div>
<p> goodbye </p>
```

CSS:

```
div        { background-color: white;  border: 2px dashed black;  padding: 1em; }
body > div { float: right; }
div div    { background-color: cyan;  margin: 1em;  text-align: right; }
#div       { text-decoration: underline; }
.div .div  { border: 2px solid black; }
p          { border: 2px solid black;  clear: both; }
```

1. **HTML / CSS Tracing (writing space)**

2. PHP

Write the PHP code for **two partial web pages** for ordering food from an online store. **The first page you will write is a form named `order.php`** that allows the user to choose what kind of food to buy and how many to buy. In your form, include a drop-down menu of food items available. Include a text box for the user to enter a quantity of the item to purchase (2-characters-wide), and an "Order" button to submit the form. The form should look like this:

Food item: milk ▾
Quantity: 10
Order

The food items to list should be based on what JPG food images are available in the current directory. For example, if the current directory contains `apple.jpg` and `steak.jpg`, then "apple" and "steak" appear in the drop-down list.

Write the portion of `order.php` that would appear between `<body>` and `</body>`. You don't need to write any CSS.

The second page you will write is named `order-submit.php`. The form in `order.php` submits its data as a POST request to `order-submit.php`. The output of `order-submit.php` is an HTML page fragment, a single paragraph indicating information about the order as described below. (Do not use `print` or `echo`.)

The store's current inventory is stored in a file named `inventory.txt` on the server in the current directory. Each line of the file represents one item available in the store, its quantity available, and its price per unit, separated by tabs. Assume that the file exists, that its contents are valid, and that there are no duplicates. Here is an example inventory:

```
apple    4     1.00
chicken  1     3.25
cookie   38    0.25
milk     9     4.50
tomato   27    0.50
```

In general your task is to look up the price per unit of the item the user is ordering, and use this price to compute the total order cost. For example, if the item costs $0.50 and the user orders 7 of them, the total order is 7 * 0.50 = $3.50.

The page's output in the general successful case is to inform the user that the order was successful, display the total order cost (you do not need to round it), and show the user a series of images representing what was ordered. For example, if the user orders 4 apples, your output should display 4 copies of `apple.jpg`.

Order successful! $4 is your total price. Here is what you ordered: 🍎 🍎 🍎 🍎

Here is another output from ordering 15 of the item 'cookie'. (the output wraps to the next line in the browser)

Order successful! $3.75 is your total price. Here is what you ordered: 🍪 🍪 🍪 🍪 🍪 🍪 🍪 🍪 🍪 🍪 🍪 🍪 🍪 🍪 🍪

The store is only able to complete an order if that food item is in the inventory and the store has enough of that item in stock. For example, if the inventory matches the above text file (which has 9 'milk' in stock), and the user tries to order 10 of 'milk', the following error should be displayed:

Sorry, we don't have 10 of 'milk' in stock.

The items in this inventory may overlap with the images that were listed previously, but there might be some images that don't have representation in the inventory text file and vice versa. If an item is not present in the inventory file, assume that its quantity available in stock is 0 and display the same sort of error message.

You do *not* need to modify the `inventory.txt` file or update its quantity of the item being ordered.

If the food item or quantity parameters are not passed, issue an HTTP 400 error. If they are present, you may assume that the food item value passed is a string and the quantity value passed is a positive integer.

Write your answer on the next page.

2. **PHP (writing space)**

2. **PHP (writing space)**

3. JavaScript / DOM

Write the **JavaScript code** to add behavior to the following page for finding palindromes. A *palindrome* is a word that is spelled the same forward as backward, such as "madam" or "Anna". The page UI allows the user to type a phrase into a text box. The user can click a "Find Palindromes" button to find palindrome words in that phrase. Match case-insensitively; for example, "rotOR" is a palindrome. You may assume that words in the phrase are separated by single spaces and contain only letters. A one-letter word such as "I" is defined to be a palindrome.

Each palindrome found should be inserted as a bullet into a list with the `id` of `palindromes`. Every other palindrome (the first, third, fifth, etc.) should be given a gray background color of #CC0000. Underneath the list of palindromes you should display text such as "5 total palindrome(s)" in the `div` with `id` of `count`.

The user can optionally specify a minimum and maximum word length by typing integer values into two text boxes with `id` of `min` and `max` respectively. If a minimum is specified, you should include only palindrome words that contain at least that many letters inclusive. If a maximum is specified, you should include only palindrome words that contain at most that many letters inclusive. If the min or max is left blank, the length is unbounded in that direction. For example, a minimum of 3 and a blank maximum finds all palindromes of at least 3 letters. You may assume that the text typed in these boxes will either be blank or a valid non-negative integer, and that max will be ≥ min.

The code should work for multiple clicks of the button. On each click it should clear any previous found information.

You may assume that Prototype and Scriptaculous are included in the page.

```html
<h1> Palindrome Finder! </h1>
<div> Phrase: <input id="phrase" type="text" size="70" /> </div>
<div> Length: <input id="min" type="text" size="2" /> to
              <input id="max" type="text" size="2" /> </div>
<div> <button id="find"> Find Palindromes </button> </div>
<ul id="palindromes"></ul>
<div id="count"></div>
```

These screenshots show the initial state, and after phrases have been typed and "Find Palindromes" is clicked.

Write your answer on the next page.

3. **JavaScript / DOM (additional writing space)**

4. SQL

Write an SQL query to search the `imdb_small` database for the id and last/first name of all directors who have made at least two films and has covered at least two different genres. Show the directors in alphabetical order by last name ascending, breaking ties by first name in ascending order. Each director should be listed only once.

A genre is a category of film such as Action or Drama. *For this problem, assume that a given movie is considered to be part of exactly one genre*, corresponding to one row in the `movies_genres` table for that film. Recall the tables:

actors

id	first_name	last_name	gender
433259	William	Shatner	M
797926	Britney	Spears	F
831289	Sigourney	Weaver	F
...			

movies

id	name	year	rank
112290	Fight Club	1999	8.5
209658	Meet the Parents	2000	7
210511	Memento	2000	8.7
...			

roles

actor_id	movie_id	role
433259	313398	Capt. James T. Kirk
433259	407323	Sgt. T.J. Hooker
797926	342189	Herself
...		

directors

id	first_name	last_name
24758	David	Fincher
66965	Jay	Roach
72723	William	Shatner
...		

movies_directors

director_id	movie_id
24758	112290
66965	209658
72723	313398
...	

movies_genres

movie_id	genre
209658	Comedy
313398	Action
313398	Sci-Fi
...	

When run on the `imdb_small` database, your query would produce the following results (see explanation at right).

```
+-------+-----------+-------------+
| id    | last_name | first_name  |
+-------+-----------+-------------+
| 11652 | Cameron   | James (I)   |
| 15092 | Coen      | Ethan       |
| 15093 | Coen      | Joel        |
| 58201 | Nolan     | Christopher |
| 78273 | Tarantino | Quentin     |
+-------+-----------+-------------+
5 rows in set (0.01 sec)
```

- Cameron directed *Aliens* (Action) and *Titanic* (Romance).
- The Coen brothers directed *Fargo* (Drama) and *O Brother, Where Art Thou?* (Adventure).
- Nolan directed *Memento* (Thriller) and *Batman Begins* (Action).
- Tarantino directed *Pulp Fiction* (Drama), *Reservoir Dogs* (Thriller), and *Kill Bill: Vol. 1* (Action).

If you join too many tables together that are not needed for the query, you will not receive full credit.
You should solve this problem using only the SQL syntax shown in class and the textbook.

Extra writing space