CSE 154 sample midterm 1

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.

a)

```
<div id="foo">
     <div id="bar">kkk</div>
     <div class="baz">mmm</div>
     <div class="baz">ooo</div>
</div>
```

```
div div {
   margin: 40px;
   background-color: yellow;
}

div {  padding: 20px;  }

#foo { border: black 2px dashed;  }
```

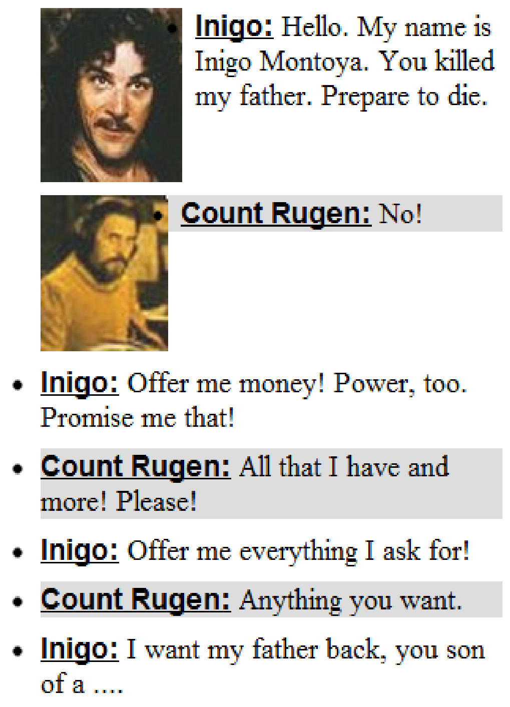
b)

```
<div id="foo””">
     <div id="bar">kkk</div>
     <div class="baz">mmm</div>
     <div class="baz">ooo</div>
     <div class="bar">hhh
          <div>ddd</div>
     </div>
     <div id="cat">^..^</div>
</div>
```

```
.baz { float: right; }
.bar { float: left; }
div { border: black solid 2px; }
#foo > div { background-color: red; }
#cat { clear: both; }
```

2. HTML / CSS Coding

Write the HTML and CSS code necessary to recreate the following appearance onscreen, between but not including the thick black lines. (No manual line breaks have been inserted into the text.) Assume that the code you're writing will be placed inside the body of the page. Part of your grade comes from choosing appropriate tags to match the semantics of the content. You should also write valid code that would pass the W3C validators, and separate stylistic information from HTML.



- **Inigo:** Hello. My name is Inigo Montoya. You killed my father. Prepare to die.
- **Count Rugen:** No!
- **Inigo:** Offer me money! Power, too. Promise me that!
- **Count Rugen:** All that I have and more! Please!
- **Inigo:** Offer me everything I ask for!
- **Count Rugen:** Anything you want.
- **Inigo:** I want my father back, you son of a ....

Some details about the desired appearance:

- All text uses default sizes and fonts, except for the characters' names which are written in a sans-serif font.
- The section of dialogue is 50px from the right edge of the page and occupies 16em in width. No manual line breaks have been inserted into the text.
- The text for each character's name is bold and underlined.
- The images come from the files `inigo.jpg` and `rugen.jpg` and are drawn at their default sizes.
- Rugen's lines are colored with a background of #DDDDDD or (red=221, green=221, blue=221). 5px separate these images from any surrounding content.
- 5px of vertical space separate each of the bulleted lines.

Mark up the text on the next page with your HTML tags. If a tag can't physically be written in the space provided, write it in the margins and draw an arrow to where it should be inserted in the text. Write the CSS styles on the page after next.

2. HTML / CSS Coding (writing space)

Inigo:

Hello. My name is Inigo Montoya. You killed
my father. Prepare to die.

Count Rugen:

No!

Inigo:

Offer me money! Power, too. Promise me that!

Count Rugen:

All that I have and more! Please!

Inigo:

Offer me everything I ask for!

Count Rugen:

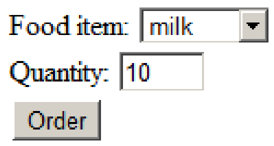
Anything you want.

Inigo:

I want my father back, you son of a ....

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:



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.*

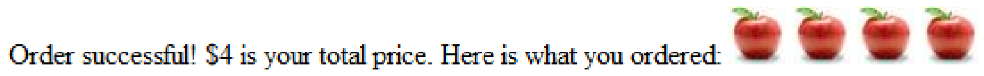
The second page you will write is named `order-submit.php`. The form in `order.php` submits its data as a POST request to `order-submit.php`. The output of `order-submit.php` is an HTML page fragment, a single paragraph indicating information about the order as described below. (Do not use `print` or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

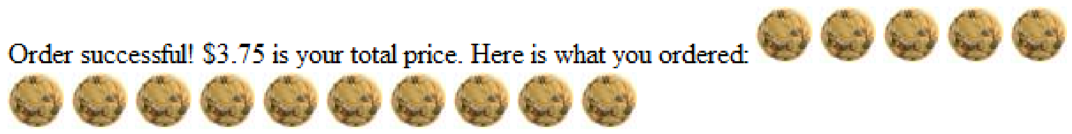
In general your task is to look up the price per unit of the item the user is ordering, and use this price to compute the total order cost. For example, if the item costs $0.50 and the user orders 7 of them, the total order is 7 * 0.50 = $3.50.

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`.



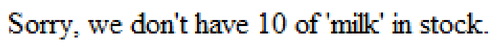
Order successful! $4 is your total price. Here is what you ordered:

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.