COMM 2555: Interactive Digital Communication / Spring 2018
Lab 9: Basic layout techniques with CSS

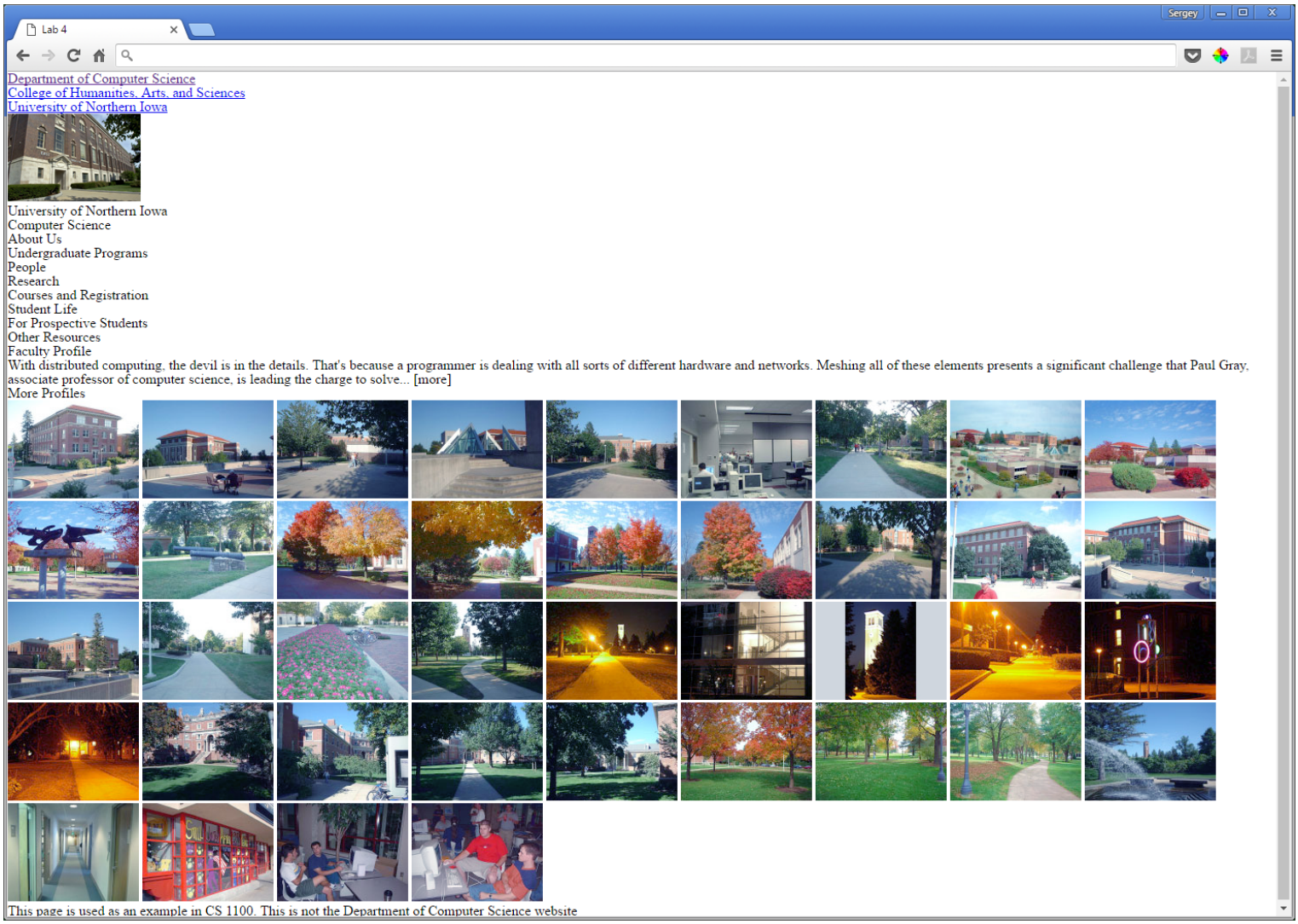
Goals

- Practice working with the CSS box model, positioning and layout

Step 1. Create your infrastructure

1.1. Create new folder, call it **lab9_your-name**.

1.2. Download the file lab9.html and open it in a browser. You should see the following:



Your task is to create a style.css file and add the styles explained in this lab. You will be adding your code to style.css only (which you need to create, of course). ***You only need to submit your style.css file.***

Step 2. Adding a background

2.1. First we add a background to our page. Add the following code to style.css, one line at a time:

1. Add the body selector with the empty curly braces
2. Add the background-image declaration ("declaration" is a property/value pair). Save file, refresh browser.
3. Do the same for the other 2 declarations. Try other values for both to see what happens.

```css
body {
    background-image: url(http://cs.uni.edu/images/layout/bg2.jpg);
    background-repeat: no-repeat;
    background-size: 100% 190px;
}
```

Step 3. Positioning elements

Make sure you have your html file open in your text editor - you need to consult it to understand what you are selecting and styling.

3.1. Let's start with the topmenu div. Let's pretend that these three links are essential at all times and, therefore, should be visible regardless of how far down the page the user scrolls.
**NOTE: When is that really the case? NOT the corporate logo or menu. It is the case for an administrative toolbar when we are editing a site - that's one of the few cases when we want part of the screen to be always visible*.

It may help to position ***and*** style an element at the same time.

3.1.1. Let's add the following code:

```css
#topmenu {
    position: fixed;
    background: #666;
    width: 100%;
    text-align: center;
    font-size: 0.8em;
    font-family: sans-serif;
    border-bottom: solid #333 2px;
}
```

Consider the code you've added:

1. The top declaration ensures our selection stays in place (try to scroll down).
2. We've given it a (neutral) background, so it looks like a ribbon.
3. With fixed positioning, we must specify the width even though the element is block-level.
4. We've added some text styling.
5. And we've added a subtle border (which adds 2 pixels to the height of our ribbon).

3.1.2. Now make it horizontal, and add some padding where needed:

```css
#topmenu li {
        display: inline-block;
        padding: 0.7em 1em 0.5em 1em;
}
```

Look carefully at the way we've specified the padding: do you remember the order? Try different values to see which one's which (top? left? bottom? right?

Why did we use 0.5em for one of the values? Why not 0.7em like the opposite side?

[.....spoiler!.....]

Because of the 2 pixel border. If we keep the values at 0.7, the text would appear to be too high (0.7em whitespace on top, 0.7em + 0.2px = approx. 0.9em on the bottom).

3.1.3. What about the links? The default color of the anchor element is blue; the default style is underlined. To change this, we need to style the a element:

```
#topmenu a {
    color: #fff;
    text-decoration: none;
}

#topmenu a:hover {
    color: lightblue;
}
```

The second rule states that when the mouse cursor is over the element, the element's color changes to light blue.

3.2. Let's take care of the photo of ITTC. It should be place in the top right corner. We'll use absolute positioning, which takes it out of normal flow; we'll also give it a border:

```
#ittc {
    position: absolute;
    right: 50px;
    top: 60px;
    border: solid #ccc 1px;
}
```

3.3. Now the name of the department + university (which is the site logo, more or less). Again, positioning and styling:

```css
#top {
    padding: 120px 0 30px 50px;
}

#top h2, #top h1 {
    color: white;
    font-variant: small-caps;
    letter-spacing: 0.1em;
    font-family: serif;
}

#top h1 {
    font-size: 3em;
}

#top h2 {
    font-size: 1.1em;
    position: relative;
    left: 30px;
    top: 5px;
}
```

Pay attention to how we group our styles to eliminate redundancy.

We use relative positioning for H2 because we want it to be positioned relative to its parent div. We could do the same by using negative margins, but this way is more elegant.

3.4. Let's take care of the columns.

Here's the idea:

1. We specify a width for both left and right divs (otherwise floating them won't work: they occupy 100% of their parent container (the body in our case) by default.
2. We float the left div to the left.
3. We float the right div to the right.
4. We add left and right margins to the center div, so that its content doesn't flow around/under the side bars.
5. Again, we add some styling

```
#left {
    float: left;
    width: 200px;
    padding: 15px;
}

#right {
    float: right;
    width: 200px;
    padding: 15px;
}

#center {
    margin: 0 230px 0 230px;
    padding: 20px 50px 0 50px;
    border-left: navy 1px dotted;
    border-right: navy 1px dotted;
}
```

3.5. Finally, let's add some style to our image gallery!

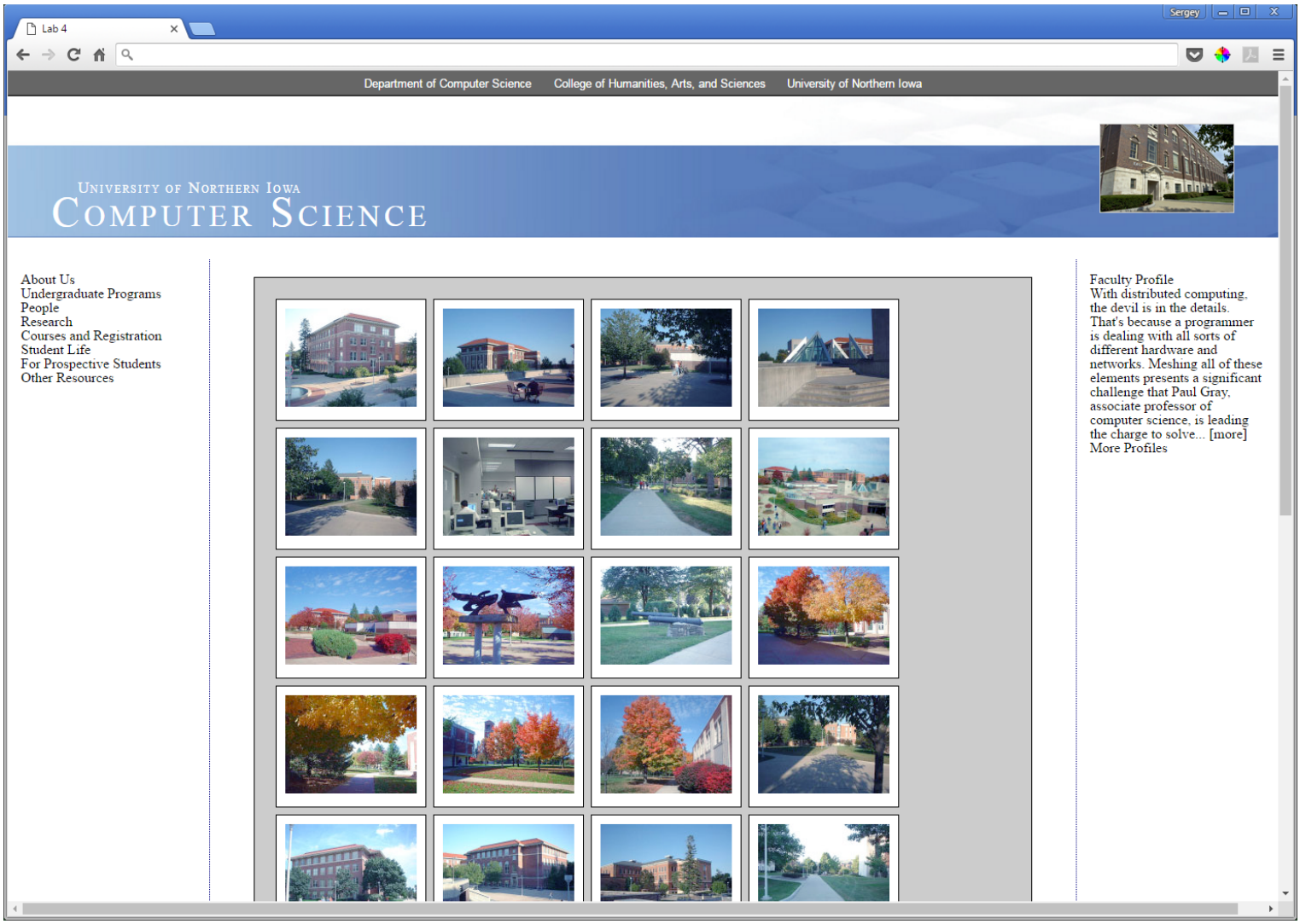
```
#gallery img {
    float: left;
    border: 1px solid black;
    padding: 10px 10px 15px 10px;
    margin: 4px;
    background: white;
    width: 150px;
}

#gallery {
    background: #ccc;
    padding: 20px;
    border: 1px solid black;
}
```

...And it doesn't work! That's because all the elements inside the gallery div are floated - that causes the gallery to collapse (it "can't see" the elements). The solution is simple: add the following declaration to your #gallery block:

```
overflow: auto;
```

That's it!



Submit your work

Save your style.css file to eLearning:
https://bb9.uni.edu > log in with CatID > our course > Course Content > Labs > Lab 9

Congratulations, you're done!