CS 2033

Multimedia and Communications

Lab 05: Learning how to create a layout using HTML and CSS; adding transitions and animations with CSS.

- Website Development -

REMEMBER TO BRING YOUR MEMORY STICK TO EVERY LAB!

The Importance of CSS

You saw last week that CSS is a useful language for styling a website in terms of colours, borders, sizes, and several other aesthetic properties. This week we are expanding on the uses of CSS beyond the noticeable appearance. The other important aspect of CSS is its structural support for building layouts using various properties such as display, position, margins, width, height, etc.

Another handy feature in CSS is its ability to easily add transitions and animations on changing properties. While this might not have the same level of importance as its structural power, it is highly valuable and appreciated by website designers. Many modern websites include some form of transitions or animations, and CSS3's support for both saves web designers time and effort.

EXERCISE 1: METHODS FOR ADDING CSS

What you'll learn in this exercise:

- Refresher on internal styles
- How to add inline styles
- How to add external styles
- Understanding when to use each option

Internal CSS

1. Navigate to your USB drive folder (F:) and into the ***cs2033*** subfolder (should have been created in a previous lab).
2. Create a folder called ***lab05*** inside ***cs2033***. Within ***lab05*** create another folder, called ***exer1***.
3. Open Brackets and create a new file (click File > New).
4. Start by saving the new file into your ***Lab05*** > ***exer1*** folder and name the file index.html
5. Add the basic webpage shell code (refer to Lab 4 for that code)
6. Add a page title in the head: "My Website – Home"
7. In the body, add an h1 header that reads "Welcome to my website!"
8. Under that header, create a paragraph that reads "I'm learning about the various methods for using CSS in a webpage. This page is styled with internal styles."
9. The HTML should look like this so far:

```
<!DOCTYPE html>
<html>
    <head>
        <title>My Website - Home</title>
    </head>

    <body>

        <h1>Welcome to my website!</h1>
        <p>I'm learning about the various methods for using CSS in a webpage.
        This page is styled with internal styles.</p>

    </body>

</html>
```

10. In the head section, create an area for CSS styles (this is a refresher from Lab 4):
    <style type="text/css">
    </style>
11. Within this style section, create a rule-set for the body tag and set the background colour to lightblue and the padding to 10px.
    body {
        background-color: lightblue;
        padding: 10px;
    }
12. After the body rule-set, add a rule-set for h1 elements to set the text colour to purple.
13. Finally add a rule-set for the paragraph (p) tag and set the font size to 18px.

```
<style type="text/css">
    body {
        background-color: lightblue;
        padding: 10px;
    }
    h1 {
        color: purple;
    }
    p {
        font-size:18px;
    }


</style>
```

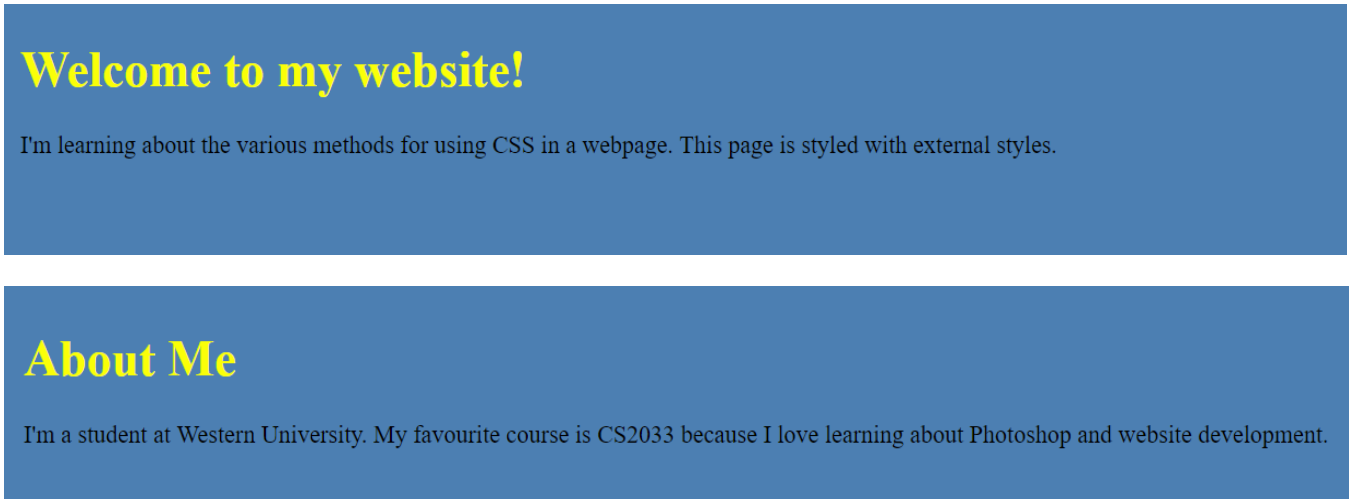
14. Save the file and open it in Google Chrome. You should see the lightblue background of the main page, the purple heading, and the black paragraph text.
15. Go back into Brackets and create another new file. Save this new one in the same folder (***lab05*** > ***exer1***) and give it the name about.html
16. Add the basic webpage shell code to this file.
17. Set the page title in the head to "My Website – About"
18. In the body of about.html, add an h1 heading that says "About Me"
19. Under the heading, add a paragraph element that says "I'm a student at Western University. My favourite course is CS2033 because I love learning about Photoshop and website development."
20. Save about.html and open it in Chrome. You should see your text there but no colours or other visible styles. That's because we only added the CSS to the first page.
21. Click into index.html. Highlight all the CSS including the opening and closing style tags. Copy that selection (Ctrl+C) and click into about.html, and paste (Ctrl+V) into the head tag. Make sure it's pasted into the correct place in the head. If you make a mistake, you can undo (Ctrl+Z) and then re-paste it in the correct place.
22. Save about.html and reload it in Chrome. Now you should see the CSS styles applied to this page just like the first page.
23. It's unlikely that you will know exactly which styles to set all at once when first making the page. Usually these styles will be changed and tweaked several times to make the page look exactly how you like.
24. Let's make a change to our CSS in about.html. The header text should be larger, so update the h1 rule-set by setting the font-size to 40px. The paragraph text can also be increased, try changing it to 26px.
25. Now go back into index.html and make the same changes (either copy the styles from about.html and paste it into index.html OR just re-type these two changes by hand in index.html). Now save both files and reload them both.
26. The header looks much better, but now the paragraph text is too big. We got too excited about increasing it! Go back into both index.html and about.html and change the paragraph text to 20px. The header styles can stay as they are since they look fine.

27. Again, save both files and reload them.
28. This looks a little better. In reality, there could be many more little tweaks like this but for the sake of this lab, let's assume this looks fine. The point to see from this is that making tweaks means changing multiple files (also remember many websites will have a lot more than 2 pages, so be happy I kept it simple for you in this lab!), which is very inefficient. You shouldn't have to copy and paste styles into multiple files like this. This method we just did is called internal stylesheets because the CSS is typed within the head of each page. The next portion of this exercise will show you how to create external stylesheets.

External CSS

As you just saw, changing styles with internal stylesheets can be tedious, inefficient, and annoying especially if the website has multiple pages and there are many tweaks to try. Instead of putting the CSS at the top of each page, it is usually more efficient to have the CSS in its own file and just link the HTML files to that CSS file. This way, changes to the styles only have to be done once and they automatically loaded in every page that links to that file. This is called external stylesheets.

1. Keep the index.html and about.html open in Brackets and in Chrome.
2. Create a new file in Brackets
3. Save it in the same folder as the HTML files (***lab05*** > ***exer1***) and name it styles.css (not .html!)
4. Click into about.html. Select all the CSS **except for** the style tags. So you will be selecting from "body {" down to the "}" bracket that closes the p rule-set. Copy this selection (Ctrl+C). Click back into styles.css and paste this code (Ctrl+V). If you accidentally included the opening and/or closing style tags in the selection, simply delete it now from styles.css
5. Now click into index.html. Select all the CSS including the opening and closing style tags and delete it all.
6. Repeat Step 6 in about.html. Both HTML files should now have absolutely no CSS in them, and styles.css should have all the CSS (again, the style HTML tags should not be included).
7. Save all the files and reload index.html and about.html in Chrome. You will see the text but now all the styles are gone. That's ok! We still need to link our HTML files to the external CSS file in order for the styles to be applied.
8. Click into index.html in Brackets. Within the head (where the style tags used to be, but are now removed), enter the following line of code:
   <link rel="stylesheet" type="text/css" href="styles.css">
9. That line links the CSS file, styles.css, to the index page.
10. Repeat step 9 in about.html to link the same stylesheet to it.
11. Save both HTML files and reload them both in Chrome. Now you should see the styles like we had before (lightblue background, purple and large font size for the header, medium font size for the paragraph). You successfully linked an external CSS file to your HTML pages!
12. To make sure our content is up to date, click into index.html in Brackets and in the paragraph text, change "internal" to "external". So the last sentence should read "This page is styled with external styles."
13. Now, we may want to tweak the styles a bit more. Go into styles.css in Brackets.
14. Change the body's background colour to steelblue instead of lightblue.
15. Change the h1's text colour to yellow and the font size to 45px.
16. Change the paragraph (p) font size to 22px.
17. Save styles.css and reload both HTML files in Chrome. See how both of them have the updated styles at once?! This is why external stylesheets are much better than internal stylesheets. All the styles are done in one place instead of copy and pasting code across multiple files. For reference, your pages should look roughly like these now in Chrome:



Inline CSS

There's a 3rd method for adding styles in a webpage but it's actually the worst option of the 3! You'll see how to do it in this portion of the lab exercise, but it's not recommended to use since it's more inefficient than the other options. External styles are in their own files so they can apply to multiple pages at once. Internal styles are within a page's header so they apply to that page alone. The 3rd option is called inline styles and they are applied to individual tags/elements in a page. Don't worry if you don't quite understand that yet. You will see shortly.

1. Click into index.html and put your cursor inside the paragraph (p) tag, directly between the p and the >. Hit the spacebar to add a space between the two characters.
2. Inline styles can be applied as an attribute on HTML tags so we are going to change the text colour of this paragraph of text individually – it won't affect any other element on this page or the about page.
3. Make sure your cursor is still immediately after the space and before the >.
4. Write the following code:
   style="color:white;"
5. Your paragraph tag should now look like this:

```
<p style="color:white;">I'm learning about
```

6. Save index.html and reload it in Chrome. The paragraph text should now show up in white.
7. Reload about.html now and you will see nothing changes.
8. Now go into about.html in Brackets and click into the paragraph tag and add a space. Type the same code as you did above to add a style to this about paragraph tag. This time, change the font colour to red.
9. Save about.html and reload it in Chrome. You'll see the paragraph in red now (I know it looks ugly and hard to read on the blue background!)

You have added inline styles to two individual elements now. As you might be able to tell, this is not ideal since the main point of CSS generally is to apply the same designs to multiple items and keep it all contained in one place. So overall, inline is the worst option for stylesheets, external is the best option, and internal is in the middle.

However, there are certain circumstances in which it's fine to use internal or even inline styles. Internal styles can be helpful if there is a single page or a specific element type that only appears on one page of a site, so it wouldn't need to be styled externally. Sometimes during testing and

tweaking styles, it's easier to set inline styles on a single element so that the stylesheets don't have to be edited repeatedly. It can save a few seconds during those testing periods to just change inline styles quickly and not have to navigate between the HTML tag and the CSS in different places. However, once the testing is done, it is best to convert it back to an internal stylesheet, or better yet, an external stylesheet.

EXERCISE 2: CSS LAYOUTS

CSS is important not only for visual, aesthetic appeal, but also for layout structure. There are many ways to create a layout using CSS. The key that is common to all approaches is the div element. An HTML div is essentially an empty panel that can be given any styles and nested within other divs or other container elements. While several other elements share these properties, most are restricted in certain ways or have to be modified before they offer the same flexibility as the div does naturally. In this exercise, we will combine styles from a provided CSS layout file and your own added styles to build a structured website without tables!

1. Open http://www.csd.uwo.ca/~bsarlo/cs2033b/labs/Lab05/exer2/template.html in Chrome. You don't have to download it, just open it and look at the different sized elements. You will be using these divs to create a website structure. Each row/section of website content will be distributed into one of the sets of the div sizes (100%, 50%, 33.3%, or 25%) depending on how many columns of content are needed for that section.
2. Create a folder in ***lab05*** called ***exer2***.
3. Open http://www.csd.uwo.ca/~bsarlo/cs2033b/labs/Lab05/exer2/, download ***exer2_shell.html*** and ***layout.css***, and save them into ***lab05*** > ***exer2***.
4. Create a folder in ***exer2*** called ***images***. Download all the images found in the exer2 > images folder on the server and move them into ***lab05*** > ***exer2*** > ***images***.
5. In Brackets, open layout.css and exer2_shell.html. Re-save exer2_shell.html as ***exer2_complete.html***.
6. Have a brief look at layout.css, particularly the last 4 selectors (.i1, .i2, .i3, and .i4). You will not add/edit code in this file; it is provided as is.
7. The exer2_complete.html file doesn't have much yet, but it has the layout.css attached to it for the different sized column elements, and it has basic HTML tags to get you started. The following steps in this exercise will provide you with HTML or CSS code snippets that you will enter in this file (HTML will go within the wrapper tag in the body, and CSS will go within the style tag within the head). Remember: do not change layout.css at any time.
8. Save the file after each step and open/refresh it in the browser to see the changes.
9. First step is to set the whole page's background colour to tan. Add this CSS rule-set within the style tag:
```
body {
    background-color:tan;
}
```
10. Let's add the top banner to the website, which will use the 100% width block. Add this HTML within the wrapper tag:
```
<section>
    <div class="ib i1"><img src="images/banner.jpg" class="banner" /></div>
</section>
```
11. This looks good! Notice this div has 2 classes: ib and i1. The first class, ib, is our generic class for any inline-block element in the layout and does not specify its width. The second class, i1, indicates that it is a 1-block meaning it takes up the entire row. The width is assigned within the i1 class (have another look at layout.css to see this).

12. The next section of the website is the navigation buttons. We want to add 4 links so we will use the i4 class instead of i1 for each of these link containers. Enter the following HTML in the wrapper tag:
```
<section>
    <ul>
        <li class="ib i4">Home</li><li class="ib i4">About</li><li class="ib i4">Travel</li><li class="ib i4">Gallery</li>
    </ul>
</section>
```
13. In the browser, you will see the text for each of the links but they don't look like nice buttons. Using CSS we will make them look nicer. Enter these 2 CSS rule-sets:
```
ul {
    margin:0;
    padding:0;
}
li {
    background-color:peru;
    text-align:center;
    line-height:50px;
    font-size:35px;
}
```
14. That should look much better! They're still missing one thing though: a rollover effect. We'll use the :hover pseudo-class to accomplish this. Enter the following CSS:
```
li:hover {
    background-color:brown;
    color:white;
}
```
15. Now when you hover your cursor over the links, they should turn to a reddish brown colour.
16. The next section of the webpage will contain some real content. There will be 2 columns (so we'll use i2); one for paragraphs of text and one for a picture. Enter the following HTML:
```
<section>
    <div class="ib i2">text here</div><div class="ib i2"><img src="images/mountain.jpg" class="side-pic" /></div>
</section>
```
17. Open http://www.csd.uwo.ca/~bsarlo/cs2033b/labs/Lab05/exer2/content.txt in Chrome. Select and copy all the content from the file and then paste it into the new HTML where it says "text here". Add at the start of each paragraph and at the end of each. (If it helps, you can first paste it into a blank file, add the tags, and then copy that back into the appropriate place in exer2_complete.html.
18. This should show up fine in the browser, but we still want to add a couple styles to make it look more appealing. Enter the following CSS:
```
p {
    font-size:18px;
}
.side-pic {
    width:100%;
}
```
19. Now we want to add a call-to-action button to attract users to subscribe to the mailing list. To make it look nice, we will add a long picture and place the button on top of the image. This means we want a single element across this row, i1. Enter the following HTML:
```
<section id="subscribe-panel">
    <div class="ib i1"><button id="subscribe">Subscribe now!</button></div>
</section>
```
20. If you look now on the browser, all you will see is a small, boring button. We now have to add CSS to display the background image behind the button and set the size and position of it. Add the following CSS rule-sets for ID selectors:

```
#subscribe-panel {
    background-image: url("images/stonehenge.jpg");
    background-repeat:no-repeat;
    height:374px;
    line-height:374px;
    margin: 20px 0;
    text-align:center;
    position:relative;
}
#subscribe {
    background-color:azure;
    width:400px;
    height:50px;
    margin: 40px auto;
    padding:10px;
    font-size:22px;
    border: solid 1px black;
}
```

21. Again we want to apply a rollover effect to the button. Add this CSS rule-set:

```
#subscribe:hover {
    background-color: lightseagreen;
}
```

22. The next section of the webpage will have 3 side-by-side boxes each with a picture clipped to a circle frame. We will use the i3 class since there will be 3 columns in the row. Add the following HTML:

```
<section>
    <div class="ib i3"><img src="images/moonlight.jpg" class="rounded-pic" /></div><div class="ib i3"><img src="images/island.jpg" class="rounded-pic" /></div><div class="ib i3"><img src="images/rocks.jpg" class="rounded-pic" /></div>
</section>
```

23. If you look on the browser, now you see the images are tall and thin. This looks pretty nifty but it's not what we wanted. Add the following CSS to give them a circular frame:

```
.rounded-pic {
    width:80%;
    margin-left:10%;
    border-radius: 50%;
    border: solid 3px darkgoldenrod;
}
```

24. Now in the browser you should see the images appear in a circular frame. Looks great!
25. Last step of this exercise is to add a footer (bottom panel) with the copyright. This will be a single long panel so we'll use i1 again. Add the following HTML:

```
<section>
    <div class="ib i1"><p class="copyright">Tamara's Travels &copy; 2019</p></div>
</section>
```

26. And we just have to add one last CSS rule-set to this page to center the copyright info:

```
.copyright {
    text-align:center;
}
```

EXERCISE 3: TRANSITIONS AND ANIMATIONS

In this exercise you will apply transitions to several different property changes to see how they improve the overall flow and professionalism of the site. You will also create looped animations and see the difference between transitions and animations.

What you'll learn in this exercise:

- Adding transitions to style changes
- Creating animations with CSS keyframes
- Changing the animation parameters

Creating pseudo-classes and transitions

1. Open http://www.csd.uwo.ca/~bsarlo/cs2033b/labs/Lab05/exer3/ and download ***exer3_shell.html*** and save it into ***lab05***.
2. Create a folder in ***lab05*** called ***images***. Download the images ***banner.jpg*** and ***rook.png*** from the server and move them into ***lab05*** > ***images***. (Note: if you click into the rook image on the browser, it might be hard to see but it is there! It is a dark image with a transparent background).
3. In Brackets, open exer3_shell.html and save it as exer3_complete.html. Your work should now be done in exer3_complete.html and you should leave exer3_shell.html unchanged.
4. Have a look at this webpage (exer3_complete) in Chrome. You'll notice most of the page is already set up including a banner at the top, 4 category boxes in the middle, and the bottom section contains a rook image and a contact form.
5. The issue with this webpage is that it's static and boring. We want to add transitions and animations to make it look more modern and appealing.

PLEASE DO NOT CHANGE ANY OF THE PROVIDED CSS OR HTML. FOR THIS EXERCISE, JUST ADD CSS BELOW THE INDICATED LINE AND ENSURE IT IS STILL WITHIN THE STYLE TAG.

6. Save the file after each step and open/refresh it in the browser to see the changes.
7. First order of business is to change the background colour of the 4 game category boxes when the user hovers the cursor over them. We use the pseudo-class :hover to accomplish this. Add the following CSS selector:
   ```
   .category:hover {
       background-color:darkblue;
   }
   ```
8. Look at it in the browser. Move your mouse around over each of the green boxes. They should each turn dark blue when you hover over them. Cool!
9. Well, it's cool, but there's more we can do with this. Instead of an immediate change in colour, there should be a transition effect so that it gradually turns from green to dark blue over a short period (typically 1-2 seconds for most transitions). Click into the pseudo-class selector you just added and hit Enter to make a new line within it. Add a 2nd style rule to it: transition: background-color 2s;
10. Hover over each green box again. You should see that the colour change to dark blue is gradual rather than immediate.
11. We also want to move the text within each container when we hover so we will now apply another :hover selector but this time it will affect the paragraph (p) element inside the hovered element. Enter the following selector into the file. Notice that the 'p' is after the pseudo-class. This just means that any paragraph contained in the category being hovered will receive these styles.
    ```
    .category:hover p {
        top:60px;
    }
    ```

12. Go back in the browser and hover over the containers and you should see that the text now jumps down, but again this is immediate rather than gradual. Click into the selector you just entered and add the following line into it to indicate a transition.
    transition: top 2s;
13. Now when you hover you should see that both the container's background colour and the paragraph text within it are being gradually changed over a 2 second period.

Creating CSS animations

Animations are similar to transitions, but they can run independently of pseudo-class selectors like :hover. If you want an object to change its colour, border, or position without having to be clicked or hovered to trigger the change, then animation is the best way to handle this.

In this portion of the exercise, we will create an animation on the rook image to make it slide back and forth horizontally.

1. The first step is to create the keyframes structure. This is a special type of CSS structure used solely for animations. Copy the following lines into the CSS section (under the pseudo-class selectors you added for the transition portion).
   ```
   @keyframes moveRook {
      0% { left:0px; }
      100% { left:600px; }
   }
   ```
2. The above code creates an animation, called moveRook, that changes the left property from 0 to 600 as the animation runs. However, we have no yet applied this animation to our rook nor indicated any other information about the animation (i.e. length, number of loops, etc.).
3. We will now apply this animation to the rook element. The rook has an ID of "rook" so we'll access it from that ID. Copy the following line into the CSS:
   ```
   #rook {
      animation: moveRook 3s;
   }
   ```
4. In the browser you should see the rook slide across to the right and then reset to its original position where it will stay. We want the animation to loop continuously so we must add the word infinite into the animation rule after "3s", i.e.
   animation: moveRook 3s infinite;
5. Now the rook follows this animation on loop. The animation uses an ease by default which is why it accelerates at the start and decelerates at the end and the movement in the middle is faster than the movement at the extremes. Add the word "linear" within the same style rule line to remove the easing effect.
6. You should now see in the browser that the rook now moves along infinitely on a linear timeline. The last thing we want to do is make it bounce back when it reaches the end rather than reset to the left position. Animations can be given a direction of normal (what we currently have here), reverse (animating from the end point to the start point), or alternate (swapping between forwards and backwards order). In this case we want to alternate the order so add the word alternate into the same style rule.
7. The rook should bounce back and forth just the way we wanted!

Additional pseudo-class transitions

Hovering over an element is the most common pseudo-class selector but there are several other kinds, including **focus** (element is selected like an input textbox that is clicked into, **active** (element is active – user is actively clicking on it), and **visited** (links you previously visited). You've seen the visited selector in use in many websites. Have you noticed that links turn from blue to purple after you click on them? That's where the :visited pseudo-class selector applies.

We're going to *focus* on the :focus selector for the last portion of this exercise. The webpage has a contact form at the bottom so we will use :focus on each of the inputs and apply a transition to them.

1. Add the following CSS to change the styles of inputs that during their focus states.
   ```
   input:focus {
       border: solid 1px blue;
       margin-left:20px;
   }
   ```
2. Go back to the browser. Click into each of the text fields to see what happens. Notice that the first 3 inputs will be shifted and given a blue border while focused, but the bottom input does not. That's because it is a textarea, not a regular text input (different HTML tag so the CSS does not apply the same).
3. If we wanted to apply a focus style set just to the textarea, we would need to add:
   ```
   textarea:focus {
       border: solid 1px blue;
       margin-left:20px;
   }
   ```
4. However, this seems redundant to have 2 consecutive rulesets that are identical but that apply to 2 slightly different tags. Instead of keeping these separate, we can merge them by simply combining the 2 selectors with a comma. You should now have one sole focus ruleset, as follows:
   ```
   input:focus, textarea:focus {
       border: solid 1px blue;
       margin-left:20px;
   }
   ```
5. Now when you click (focus) the textarea, it behaves the same as the text inputs above it.
6. The last thing we need to do is apply a gradual transition to these styles. Again, we want this to work on both the inputs and the textarea so we will combine them into on grouped selector with a comma.
7. We also want both the border and the margin styles to be gradual so we use the word "all" within the transition to indicate that all style changes are included in the transition. In the previous transition examples, we were only changing one style so this "all" wasn't needed.
   ```
   input, textarea {
       transition: all .5s;
   }
   ```
8. Now in the browser, click into each of the 4 form elements (3 textboxes and 1 textarea) and see how the styles are gradual. The border transition is hard to see but the margin shift should be clear. You now know how to create transitions on pseudo-classes and animations.

This concludes this lab session. Call your TA over to check your work and receive your mark for this lab.

REMEMBER TO REMOVE YOUR MEMORY STICK FROM YOUR MACHINE AND PUT IT IN YOUR BACKPACK! (don't forget it)! ☺