Faculty of Arts & Sciences
Department of Computer Science
CMPS 278-Web Programming and Design
Spring 2016/2017- HW2
Due date: Saturday Feb 25, 2017.

Problem 1: Movie review (HTML and CSS.)

You will create files for a fake movie review web site named Rancid Tomatoes for the film *The God Father: Part II*. Turn in the following files:

- movie.html
- movie.css, the style sheet for movie.html

You will recreate the page below. We do not expect you to produce a pixel-perfect page that exactly matches this image. But your page should follow the styles specified below and match the look, layout, and behavior shown here as closely as possible.

We will provide you a **skeleton of movie.html** (in problem3 folder) with the page contents, but no page sections or styles. The only modifications you should make are to divide it into sections using `div`/`span` tags, and add `id` and `class` attributes. You are also allowed to add the HTML necessary to enable the "favorites icon" or "favicon" specified next and the meter tag for ranking scalable measurements. The screenshots in this document were taken on Windows in Firefox, which may differ from your system.

All **images** are provided in the folder resources.

banner.png

godfather.jpg

background.png


fresh.gif


rotten.gif


critic.gif


Al.jpg


deniro.jpg


duvall.jpg

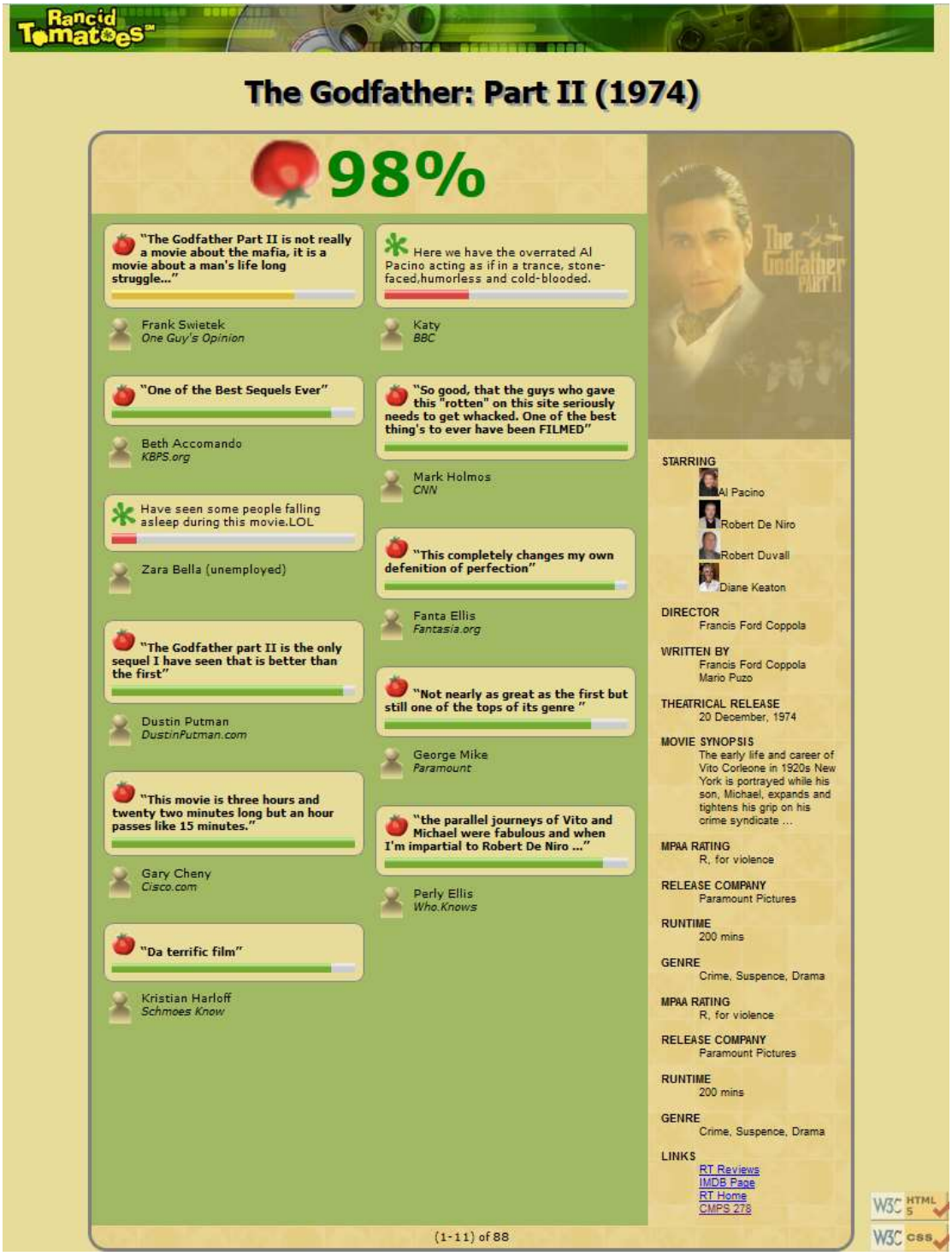

diane.jpg

The page's title is **The Godfather: Part II - Rancid Tomatoes**. The page has a "favorites icon" ("favicon") of fresh.gif.

The page background color is #E8DC9B. Page text uses 8pt font, using Verdana, Tahoma, or any sans-serif font on the system. The page body has *no margin or padding*, so its contents stretch to the very edge of the browser window.

The top of the page is an **image banner**. The center of this banner is banner.png. Each image is 50px tall. Underneath the image banner is a centered heading containing the movie name and year in a 24pt bold font. The preferred fonts for this heading are Tahoma, Verdana, or any sans-serif font available on the system. The text in the header has a "shadow" located 3px right and 3px down from the original text, using the color #999999.

Below the main heading is the page's **overall content area**, with an overall 98% rating for the film, several critics' reviews, and an overview of the film at right. Taken together this content occupies 800px in width and is centered horizontally within the page. If the page resizes horizontally, this 800px section should move dynamically so that it remains centered horizontally on the page. This overall section has a 4px gray solid border with a 20px round radius and should be sized large enough to contain all of its contents.



In the overall area there is a 585px-wide left section for the "fresh" 98% rating and the **critics' reviews** of the film. The section is topped by a smaller section containing a large "fresh" image (fresh.gif) (image dimensions are 75px*75px), vertically aligned to be even with the bottom of the text around it and **should keep rotating in a speed of 1 second**. Behind this the image background.png repeats across the entire length of the section. Each of these images is 83px tall. This is followed by the 98% overall rating for the film, which is shown in a 48pt green bold font *(Image and rating are centered).*

Below the 98% overall rating, in the **critics' reviews** section there are two **columns of reviews (**The background color of the **critics' reviews** section is #A2B964). . The columns each occupy 47% of the width of the overall left-center section of the page. There is a horizontal spacing of 2% between the columns and neighboring content.

Each **review** is a box with a quote about the movie, in bold 8pt font, except the rotten reviews which are not in bold. The quote box has a background color of #E8DC9B, and a gray border, 2px thick with a 10px round radius. 8px separate the quote box's content from its border. Each box has an icon (fresh.gif or rotten.gif) for whether the reviewer liked or disliked the movie on the left side of the quote box, with 5px separating it from text to its right. For fresh and rotten rating the text wraps around the images as needed. Numerical ranking is represented with a scalar measurement ("i.e. meter tag") where reviews with less than 50% ranking their meters appears in red, between 50% and 80%, the meter appear in orange, above 80% the meter appears in green.

The reviewer's personal information follows under the quote box, including: the reviewer's name; and the publication in italic. A reviewer icon (critic.gif) is shown to the left of the text, with 5px of horizontal space separating it from the text. There is 20pt of vertical space between reviews. *(Hint: Paragraphs in movie reviews should be made large enough to contain all their content, including any floating content. See textbook section 4.3.3 on making contents fit into a container.)*

The 11 reviews should be as shown.

To the right of the critics' reviews is an o**verview** section of the page with a list of information about the movie, with a background image of background.png. This section is 214px wide, with 10pt between the edge of the section and the text of the list. Its text appears in an 8pt font of Arial or any sans-serif font available on the system. You may assume that the overview section will always be taller than the section of reviews to its left.

The top of this section includes a **godfather.jpg** image (as shown in the figure above). **This image should be animated using different levels of opacity**. It also a definition list (using `dl`, `dt`, and `dd` elements) about the movie such as its stars and director. **In the starring part, a bullet image should be added before each star's name**. Each term is bolded and has 10pt of vertical separation between it and the element that precedes it. The bottom of this section contains a list of links about the movie. The list should be shown without bullets or indentation.

Below the reviews is a bar of centered text indicating reviews "(1-11) of 88", with a background image of background.png. It is placed directly next to surrounding content. 5px separate the edge of its text and the element's own outer edge.

The page's bottom right corner has links to the **W3C validators**. These images should remain fixed in the page's bottom-right corner, even after scrolling. The images should appear as 50% opaque, halfway transparent. For full credit, your page must use **valid HTML5** and successfully pass the W3C HTML5/CSS validators.

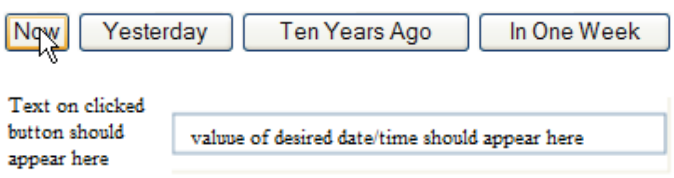
Part of your grade comes from expressing your CSS concisely and without unnecessary or **redundant** styles.

We strongly recommend that you use the **Firebug** add-on for Firefox to inspect elements.

Problem 2: Intro to JavaScript

1) Create a web **`guess.html`** that makes use of a JavaScript program **`guess.js`** that you have to write to generate a random integer between 1 and 20 then the program prompts the user to input a guess number. If the user input matches with randomly generated number, the program will display in an **alert window** the message "**Good Work, the number ix X and you guessed it after Y trials**" (where X is the random number and Y is number of user guesses". Otherwise the program will keep prompting the user for another input with a hint such as "the number is larger" or "the number is smaller" than the user guess, This hint should appear as default value in the text field of the new prompt so the user can override it. If the user was not able to hit/guess the randomly generated number after 6 guesses then the program should stop after alerting the user with ""**Bad luck, the number was X and it seems you don't know binary search**".

2) Create a web page **`dates.html`** that contains four HTML buttons. Add **`dates.js`** to your web page such that when each button is clicked, it should cause the text appearing before the text field to change to the text appearing on the button and the value of the text field to change to the desired day and time. Create a Now button that for the current time and date, a Yesterday button for the time and date 24 hours ago and other two buttons for the time and date ten years ago and one week from today.



3) Create a web page **`leap.html`** and write a JavaScript function in **`leap.js`** that can be used to determine whether a given year is a leap year in the Gregorian calendar. This function should be used in your web page which should contain a button (check year), text field, and a text area HTML element with an id="**result**". When the button is clicked, the function should read the value of the year entered in the text field then determine if the year is leap. The result should appear in one line in the text area element such as "**yyyy is a leap year**" or "**yyyy is not a leap year**". If the text field has no value or it is not composed of four digits then you should add to the text area a warning line in capital letter such as "THE ENTRY IS INVALID". The text area should not be overridden but it should be appended with a result line after every click on the button.

To determine whether a year is a leap year, follow these steps:

Step-1 : If the year is evenly divisible by 4, go to step 2. Otherwise, go to step 5.
Step-2 : If the year is evenly divisible by 100, go to step 3. Otherwise, go to step 4.
Step-3 : If the year is evenly divisible by 400, go to step 4. Otherwise, go to step 5.
Step-4 : The year is a leap year (it has 366 days).
Step-5 : The year is not a leap year (it has 365 days).

4) Create a web page **`creditcard.html`** that allows a user to enter a credit card number in a text field in the following format "dddd-dddd-dddd-dddd" (where d is any digit) subject to the below constraints, a button validate, and a P html element with id=results. Write a java script program **`creditcard.js`** to validate credit cards with a simple function that returns `true` or `false` and called after a click on a button "validate". Your program should display the result the P element whose Id = result. If the function return true then you append to the paragraph a line with "Card number dddd-dddd-dddd-dddd is valid" with green text. Otherwise, "Card number dddd-dddd-dddd-dddd is invalid". The result paragraph should be appended every time a card is validated with the right message. Every message should appear in a separate line within the paragraph. Here are the rules for a valid number:

- Number must be 16 digits, all of them must be numbers
- You must have at least two different digits represented (all of the digits cannot be the same)
- The final digit must be even
- The sum of all the digits must be greater than 16

The following credit card numbers are valid:

- `9999-9999-8888-0000`
- `6666-6666-6666-1666`

The following credit card numbers are invalid:

- `a923-3211-9c01-1112` *invalid characters*
- `4444-4444-4444-4444` *only one type of number*
- `1111-1111-1111-1110` *sum less than 16*
- `6666-6666-6666-6661` *odd final number*

5) Create a Web page named `grades.html` and its `grades.js` that can be used to compute your grade for CMPS 278. The page should allow the user to enter grades for home works, project, quizzes, participation, and final exam as illustrated below. It should also compute and display the averages for each grade category, the overall average for the course, and the raised grade. Recall from the class syllabus that grades are weighted as follows:

Assignments	20%
Project	8%
QUIZ1	14%
QUIZ2	14%
QUIZ1	14%
Final exam	25%
Attendance and Participation	5%

It is left to you to use the appropriate HTML and CSS to provide an appealing interface. However, a simple basic interface for your page might be illustrated as following (note that below, calculation might not be accurate):

Assignments	: 95 80 60 100	**Homework average**:	83.75%
Project	: 90		
Quizzes	: 80 100 90	**Quiz average**	: 91.25 %
Participation	: 88		
Final exam	: 83	**Course overall average**:	86.5
Raise coefficient	: 25	**raised grade**	: ??

You can assume that the texts are labels and grades/averages (or any digits) are text fields. Also the text fields for homework average, quiz average, course overall average, add raised grade are all not editable.

Since there will be many homework and quiz grades, we require to allow the user to enter related values at the same text field each separated by a whitespace. That is, they would enter all homework grades at the assignments, all quiz grades at the quizzes.

We assume that the initial values of all text fields are set to 0.

After entering/updating the grades for the assignments, your script should compute and display the homework average, and update the course overall average and the raised grade (read about **`onblur`** event, which can be used to execute a JavaScript when a user leaves an input field **`<input type="text" onblur="myFunction()">`**).

After entering/updating the grades for the quizzes, your script should compute and display the quiz average, and update the course overall average and the raised grade.

At any time, updating any value in any text field might lead to updating update the course overall average and the raised grade.

Average of the quizzes and home works should be computed as described next. To compute the average of the home works just sum the grades and divide the results by the number of home works. At max you can have **six** home works and at least **four**. The quizzes must be 3 and their weights are as indicated in the table.

Note that to compute the weighted average, you simply have to multiply each component of the grade by its percentage of the overall grade and add them.

Example of the grade computation using the values in the above illustration:

Homework average is computed as following: (95 + 80 + 60 + 100)/4 = 83.75%
Quiz average is computed as following: (80 * 0.14) + (100 * 0.14) + (90 * 0.14) = 37.5 / 42 = 90%
overall average is computed as following: 83.75 * 0.2 + 90 * 0.08 + 90 * 0.42 + 88 * 0.05 + 83 * 0.25
= 16.75 +7.2 + 37.5 + 4.4 + 20.75 = 86.6%

raised grade = overall average + (100 - overall average)* (participation/100)*(raise/100)

Submission

You will submit via Moodle a compressed file called **cmps-278.hw2.USER-ID**, where USER-ID is your user-id. The compressed file is an archive of a directory called cmps-278.hw2.USER-ID that contains two subfolders problem1and problem2. Put in these folders both `.html` files and your `.css` style sheets for your solution along with the provided images (it is recommended that for problem 2 use JS to set the real time styling).

Please do not place a solution to this assignment online on a publicly accessible web site. Doing so is considered a violation of the textbook academic integrity policy