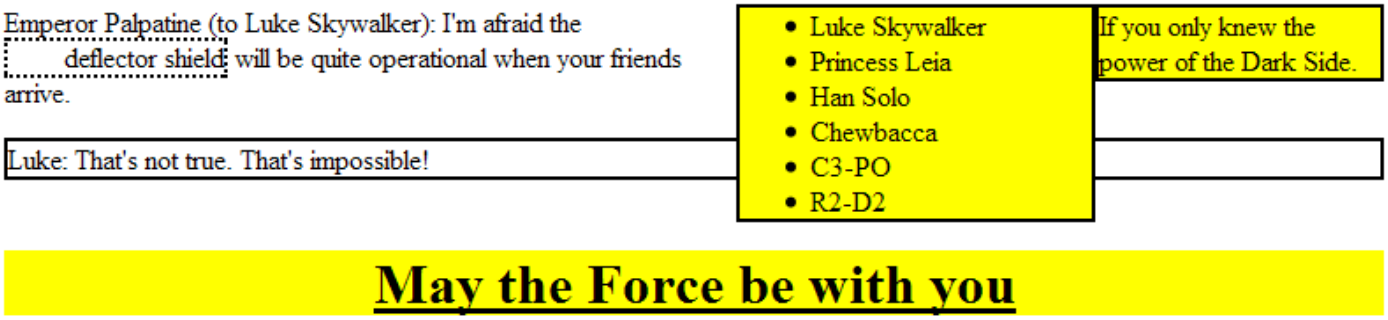
[edit] Question 1 (HTML/CSS Interpretation)



20 points total

- 7 two floating sections
  - 2 float to right
  - 2 correct L/R order
  - 2 border and background color
  - 1 width (approximate), vertical alignment
- 4 Emperor quote
  - 1 text wraps around floating paragraphs
  - 2 deflector shield dotted border
  - 1 deflector shield left padding
- 3 Luke quote
  - 2 has solid border
  - 1 border goes under/past the floating content
- 5 May the Force be with you
  - 1 centered
  - 2 underneath floating stuff (clear)
  - 2 background color
- 1 misc. point for any other random shit they fuck up
  - -1 point each for any other misc. stuff they do wrong that isn't listed above

.

[edit] Question 2 (PHP)

```
<?php
if (isset($_REQUEST["student"])) {
    $student = $_REQUEST["student"];
    $filename = "students/$student/asciimation.txt";
    if (file_exists($filename)) {
        $text = file_get_contents($filename);
        $frames = explode("=====\n", $text);
        $frame = 0;
        if (isset($_REQUEST["frame"])) {
            $frame = (int) $_REQUEST["frame"];
        }

        if ($frame < count($frames)) {
            header("Content-type: text/plain");
            print $frames[$frame];
        }
    }
}
?>
```

20 points total

- 6 student
  - 2 grabs from REQUEST
  - 2 checks whether student is set
  - 2 creates proper file name from student name
- 6 frame
  - 2 grabs from REQUEST
  - 2 default of 0 if not set
  - 2 no output if frame >= array length
- 5 file/frames
  - 2 checks whether file exists
  - 1 reads entire file properly
  - 2 splits properly on =====\n
- 3 output
  - 1 header content-type text/plain
  - 1 prints frame
  - 1 <?php ?> surrounding tags

.

[edit] Question 3 (JS/DOM)

```
var answer;
var guesses = 0;

document.observe("dom:loaded", function() {
    $("makeguess").observe("click", makeGuessClick);
    answer = parseInt(Math.random() * 100) + 1;
});

function makeGuessClick() {
    guesses++;
    var guess = $("number").value;
    var li = document.createElement("li");
    li.innerHTML = guess;

    if (guess > answer) {
        $("result").innerHTML = "Too high!";
        li.addClassName("high");
    } else if (guess < answer) {
        $("result").innerHTML = "Too low!";
        li.addClassName("low");
    } else {
        $("result").innerHTML = "You got it right in " + guesses + " tries!";
        $("makeguess").disabled = true;
    }

    $("guesses").appendChild(li);
}
```

20 points total

- 3 window onload
  - 1 window.onload or document.observe handler declared properly
  - 2 onclick handler attached to 'makeguess' button properly
- 4 picks a random answer from 1-100
  - 2 attempt
  - 2 correct (correct range, scales up properly, etc.)
- 3 game play
  - 1 grabs user's guess properly out of 'number' field
  - 2 too high/low clues in 'result' span
- 5 end of game
  - 2 counts guesses properly
  - 1 puts correct message into 'result' (you got it right!)
  - 2 disables 'makeguess' button
- 5 li for each guess
  - 1 creates/puts proper text in it
  - 2 CSS class
  - 2 puts onto page (appendChild)

.

[edit] Question 4 (Ajax/XML)

```
document.observe("dom:loaded", function() {
    new Ajax.Request("q5.xml", {
        method: "get",
        onSuccess: ajaxSuccess
    });
});

function ajaxSuccess(ajax) {
    var moves = ajax.responseXML.getElementsByTagName("move");
    for (var i = 0; i < moves.length; i++) {
        var animal = moves[i].getAttribute("animal");
        var move = moves[i].firstChild.nodeValue;

        var critters = $$("." + animal);
        for (var j = 0; j < critters.length; j++) {
            var x = parseInt(critters[j].getStyle("left"));
            var y = parseInt(critters[j].getStyle("top"));
            if (move == "N")      { y -= 20; }
            else if (move == "S") { y += 20; }
            else if (move == "W") { x -= 20; }
            else if (move == "E") { x += 20; }
            critters[j].style.left = x + "px";
            critters[j].style.top = y + "px";
        }
    }
}
```

20 points total

- 3 window onload
  - 1 window.onload handler attached properly
  - 1 Ajax request attempt
  - 1 method "get"; attaches onSuccess handler properly
- 1 Ajax onSuccess handler header (including ajax parameter)
- 6 XML parsing
  - 2 grabs all "move" elements and loops over them
  - 2 grabs attributes e.g. "animal"
  - 2 grabs .firstChild.nodeValue inside each move
- 4 grabs all proper animal spans using $$, and loops over them
  - 1 attempt
  - 3 correct (remembers '.', understands that $$ returns an array, etc.)
- 6 moves each animal
  - 1 attempt: if/else involving N/S/E/W and x/y +- 20px
  - 2 gets old position correctly (getStyle left/top)
  - 1 sets new position correctly, attempt
  - 2 sets new position correctly, correct (appends 'px', no setStyle, etc.)

.

[edit] Question 5 (SQL)

```
SELECT DISTINCT a.first_name, a.last_name
FROM actors a
     JOIN roles r1 ON r1.actor_id = a.id
     JOIN movies m ON m.id = r1.movie_id
     JOIN movies_genres mg ON mg.movie_id = m.id
     JOIN roles r2 ON r2.actor_id = a.id
     JOIN roles r3 ON r3.actor_id = a.id
WHERE r1.movie_id <> r2.movie_id
     AND r1.movie_id <> r3.movie_id
     AND r2.movie_id <> r3.movie_id
     AND mg.genre = 'Action'
ORDER BY a.last_name, a.first_name;
```

25 points total

- 4 SELECT
  - 2 columns first_name, last_name (probably requires the student to give names to their tables in the query)
  - 2 DISTINCT (no duplicates)
- 6 FROM/JOIN (choice of tables)
  - 2 has at least 3 'roles' records
  - 2 has at least 1 'movies' record
  - 2 has at least 1 'movies_genres' record
    - -2 for each extra record added unnecessarily if it screws up the query; -0 if it doesn't
- 8 WHERE/ON (constraints)
  - 2 actor <-> roles by id/actor_id
  - 1 role <-> movie by movie_id/id
  - 1 movie <-> genre by id/movie_id
  - 2 three roles are different
  - 2 at least one role has genre of 'Action'
  - -2 for each extra incorrect constraint if it screws up the query; -0 if it doesn't
- 2 ORDER BY
  - 1 last_name
  - 1 first_name and order