

UNIVERSITY OF TORONTO
Faculty of Arts and Science
Midterm Exam – October 2006
CSC 309 H1 F
Instructor – Eyal de Lara
Duration – 1 hours

Examination Aids: One single-sided hand-written 8.5"x11" page containing notes

NAME _____

STUDENT NUMBER _____

PART I (Short Answers): _____ /40
PART II (Programming): _____ /60
Total: _____ /100

Part I of this examination is a series of short-answer questions.

Part II of this examination is a programming exercise.

Answer all questions in the spaces provided on this examination paper. There is no need to use more space than is provided. If you must, then use the back of the examination paper and so indicate in your answer.

General Advice:

- Skim through the entire exam before beginning your detailed work, to get a sense of where best to spend your time; if you get stuck on one question, go on to another and return to the difficult question later.
- Show your work, not just the final answer. Partial credit will be granted in cases that demonstrate correct reasoning, even if an error leads to an incorrect final result.
- For the programming exercise, if unsure of API details or language syntax details, try your best and include a comment indicating what you are attempting to achieve.

Good luck!

Part I – Short Answer Questions

1.- [10 points] Give two examples of HTML tags that were not originally meant for document formatting, but are commonly used by designers to achieve precise content layout in a page. Briefly describe how these tags can be used for formatting.

`<table>` Used to achieve precise alignment of text and other graphic objects, e.g., images.
`` Transparent images are used to set spacing between objects

2.- [10 points] Give two examples of ways in which CSS increases programmer productivity.

Can change formatting for a large number of documents by changing one common stylesheet.
Device-specific or user-specific customizations can be done through the use of different stylesheets, removing the need to maintain multiple versions of content.
Pages are smaller as there is less need to type over and over common formatting attributes.

3.- [5 points] What is the main difference between HTTP1.0 and HTTP1.1?

HTTP1.1 uses persistent connections, whereas HTTP1.0 uses a new connection for each HTTP request.

4.- [5 points] Despite its differences from HTTP1.0, HTTP1.1 remains a stateless protocol. Why is this the case?

HTTP1.1 servers do not keep client state, other than the basic TCP state that allows for persistent connections.

5.- HTTP is a text-based protocol:

a) [5 points] Why is this an advantage?

It is easy to implement.
It is easy to debug.
Possible to monitor activity by just monitoring network traffic or logs.

b) [5 points] Why is this a disadvantage?

Network coding and processing/parsing is not as efficient as that of a binary format.

Part II - Programming Question

6.- [60 points] Using JavaScript, XHTML and a provided Java Applet write a simplified virtual version of the classic arcade game Whack-A-Mole. The purpose of the original game (illustrated in Figure 1) is to hit “moles” with a hammer before they hide away. In the virtual version, users will try to click with the mouse on an image of a single mole before it moves to a different position on the screen.



Figure 1

When loaded, your application should present an interface similar to Figure 2. You can assume that the image used to draw the mole is a locally available file called mole.gif.



Figure 2



Figure 3



Figure 4

The game starts when the user presses the Start button. At this time, the mole should move to a random location on the screen. Assume that the browser window area is square of 400px by 400px. The mole will stay in this new position for a random amount of time between 1 and 2 seconds, at which time the mole will move to the next random position, and so on and forth. However, if the user manages to click on the mole before the timeout expires, the user scores a point and the mole moves immediately to the next random location. This will continue for 5 *tries*, after which the game will be over (see Figure 4). After each try, the interface should reflect the current score and try number (Figure. 3)

Additionally, when the user scores a point (i.e., whacks a mole) your application should use a provided Java Applet to play the locally available file whack.wav, which makes a funny whacking sound. The code of the applet is shown in Figure 5. You can assume that a compiled version of the Applet is locally available.

Finally, a list of methods implemented by the JavaScript built-in Math Object are provided in Figure 6 for your reference.

```
import java.applet.*;

public class PlaySound extends Applet {
    AudioClip sound;

    public void play(String s) {
        sound = getAudioClip(getCodeBase(), s);
        sound.play();
    }
}
```

Figure 5

Methods	Description
abs(x)	Returns absolute value of x.
acos(x)	Returns arc cosine of x in radians.
asin(x)	Returns arc sine of x in radians.
atan(x)	Returns arc tan of x in radians.
atan2(y, x)	Counterclockwise angle between x axis and point (x,y).
ceil(x)	Returns the smallest integer greater than or equal to x. (round up).
cos(x)	Returns cosine of x, where x is in radians.
exp(x)	Returns e ^x
floor(x)	Returns the largest integer less than or equal to x. (round down)
log(x)	Returns the natural logarithm (base E) of x.
max(a, b)	Returns the larger of a and b.
min(a, b)	Returns the lesser of a and b.
pow(x, y)	Returns x ^y
random()	Returns a pseudorandom number between 0 and 1. Example(s) .
round(x)	Rounds x up or down to the nearest integer. It rounds .5 up. Example(s) .
sin(x)	Returns the Sin of x, where x is in radians.
sqrt(x)	Returns the square root of x.
tan(x)	Returns the Tan of x, where x is in radians.

Figure 6 Methods of the JavaScript Math Object

Extra space for question 6

```
<html>
<head>
  <script type="text/javascript">
    var interval;
    var score = 0;
    var moves = 0;

    function playSound() {
      var applet = document.getElementById("sound");
      applet.play("chord.wav");
    }

    function whackmole() {
      playSound();
      score++;
      moveMole();
    }

    function moveMole() {
      var top=50+Math.floor(Math.random()*300);
      var left=50+Math.floor(Math.random()*300);
      var mole = document.getElementById("mole");
      mole.style.left = left;
      mole.style.top = top;

      var timeout = 1000 + Math.floor(Math.random()*1000);

      var s = document.getElementById("score");

      clearInterval(interval);

      if (moves < 5) {
        interval = setInterval(moveMole,timeout);
        s.innerHTML = " " +score + " out of " + moves;
      }
      else {
        s.innerHTML = " " +score + " out of " + moves + " Game Over!!!";
        endGame();
      }
      moves++;
    }

    function startGame() {
      moves=0;
      score=0;
      moveMole();
      document.getElementById("mole").onclick = whackmole;
    }
  </script>
</head>
</html>
```

```
        function endGame() {
            document.getElementById("mole").onclick = "";
        }

</script>
</head>

<body>
<p>
    Score: <span id="score">0</span>
</p>

<input type="button" onclick="startGame();" value="Start" />

<applet id="sound" code="PlaySound.class" width="0" height="0">
</applet>


</body>
</html>
```