CPSC 101: Connecting with Computer Science
Sample Final Exam, December 2012

Name: _____ Student ID: _____

Signature: _____

- You have **150 minutes** to solve the **9** problems on this exam.
- A total of **100 marks** is available. You may want to complete what you consider to be the easiest questions first!
- Ensure that you clearly indicate a legible answer for each question.
- You are allowed two pages of notes for reference ($8.5'' \times 11''$, front and back). Otherwise, no notes, aides, or electronic equipment are allowed.
- Good luck!

UNIVERSITY REGULATIONS

1. Each candidate must be prepared to produce, upon request, a UBCcard for identification.

2. Candidates are not permitted to ask questions of the invigilators, except in cases of supposed errors or ambiguities in examination questions.

3. No candidate shall be permitted to enter the examination room after the expiration of one-half hour from the scheduled starting time, or to leave during the first half hour of the examination.

4. Candidates suspected of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action:

- having at the place of writing any books, papers or memoranda, calculators, computers, sound or image players/recorders/transmitters (including telephones), or other memory aid devices, other than those authorized by the examiners;
- speaking or communicating with other candidates; and
- purposely exposing written papers to the view of other candidates or imaging devices. The plea of accident or forgetfulness shall not be received.

5. Candidates must not destroy or mutilate any examination material; must hand in all examination papers; and must not take any examination material from the examination room without permission of the invigilator.

6. Candidates must follow any additional examination rules or directions communicated by the instructor or invigilator.

P1	P2	P3	P4	P5	P6	P7	P8	P9	Total
9	15	10	8	18	12	16	9	3	100

This page intentionally left (almost) blank.
If you put solutions here or anywhere other than the blank provided for each solution, you must *clearly* indicate which problem the solution goes with and also indicate where the solution is at the designated area for that problem's solution.

1 Data Structures and Algorithms [9 marks]

1. A computer program is trying to determine whether the name of a novel appears in a list of 1,000,000 book titles that are in alphabetical order. Where should the computer look first for the name? **[3 marks]**

If the titles are not in alphabetical order—or indeed any known order—explain why looking in the location you indicated no longer has any particular advantage. **[3 marks]**

2. Choose one of: (1) a record of mating relationships in a baboon tribe, (2) the flow pattern of a river system, and (3) a corporation's employment structure. Explain how your chosen example illustrates one of the three types of data structures: networked, hierarchical, or tabular. **[3 marks]**

I choose: **(CIRCLE ONE)** baboon relationships river system employment structure

This illustrates the data structure type: **(CIRCLE ONE)** networked hierarchical tabular

because…

2 Minds & Machines [10 marks]

Researchers create a program that can detect repetitive tasks that users perform on their computers and automatically complete them. The program silently "watches" the user type and use the mouse and when it notices a task that could be automated, it pops up a message offering to complete the task for the user.

For example, if the user were deleting each `<br>` tag in an HTML document and typing a `<p>` in its place, the program might notice this after two or three of the tags were changed, pop up a message saying "Would you like me to find the remaining `<br>` tags in the document and replace them with `<p>` tags?", and if the user answers "yes", replace the remaining tags automatically. However, the program does sometimes make mistakes, overlooking tasks that it could help with or noticing and responding to "patterns" that don't really exist.

1. Argue articulately either **for** or **against** (but not both) the intelligence of this program in your own words but appealing where appropriate to Turing's, Searle's, Hawkins's & Blakeslee's, and Aamodt's & Wang's ideas. (You need not refer to all of these, but you should make compelling reference to at least one.)

Be sure to briefly but clearly articulate your definition of intelligence as part of your argument! **[6 marks]**

2. Give a brief but strong counter-argument to your argument that one or more of the authors mentioned above might provide. Cite the author(s) who you believe would take exception to your argument and describe how and why they would disagree. (Between this part and the previous, you should make compelling reference to at least two of the authors.) **[4 marks]**

3 Art and Artists [8 marks]

1. Which of these is *NOT* a reason that computer artist Vera Molnar, A. Michael Noll, Joan Truckenbrod, or Harold Cohen cited for using computers to create art? **[2 marks]**

(a) To create artistic works faster than is possible by hand.

(b) To gain more precise control over artistic expression.

(c) To explore how people perceive and understand art.

(d) To explore how artists perceive and understand art.

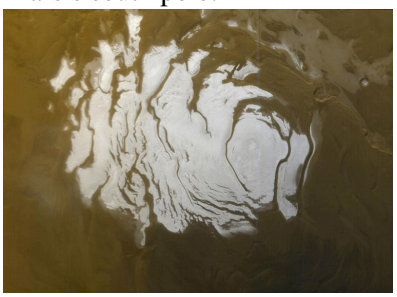
2. With respect to an artist's use of a computational tool (such as pen-based interfaces, printing, 3-D visualizations, etc.), how does the artist's need for technological expertise tend to vary with the maturity of the tool? **[3 marks]**

3. For each of the following images, indicate which of **bitmap** or **vector** image representations would have been the better choice to store the **image**—not the model or scene that created the image—the first time it was stored on a computer. **[3 marks]**

Design of the Canadian flag:

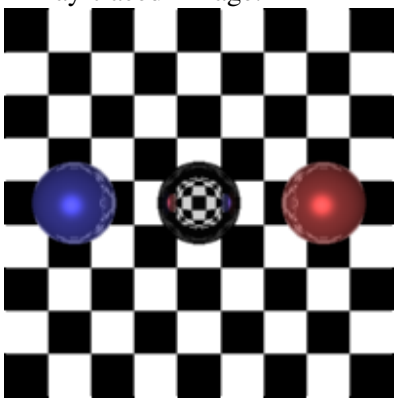
Circle one: **bitmap** **vector**

Mars's south pole:



Circle one: **bitmap** **vector**

A "ray-traced" image:



Circle one: **bitmap** **vector**

4 Machines [18 marks]

1. Is it possible to translate every valid JavaScript program into machine language? **[2 marks]**

   (a) No, because JavaScript programs execute in web browsers, not on the computer itself.

   (b) Maybe, because the human mind is capable of reasoning about programs in ways that computers are not.

   (c) Yes, because that's how they run on a computer.

2. As with computer networks, computers themselves use abstraction layers.

Indicate the correct order from highest abstraction level (closest to human thought) to lowest (simplest level of the machine) for the following jumbled steps in creating and executing a program on a computer. Put a 1 next to the highest level, a 2 next to the second highest, and so on with 5 for the lowest. **[4 marks]**

_____ A collection of gates (like NAND gates) performs logical operations.

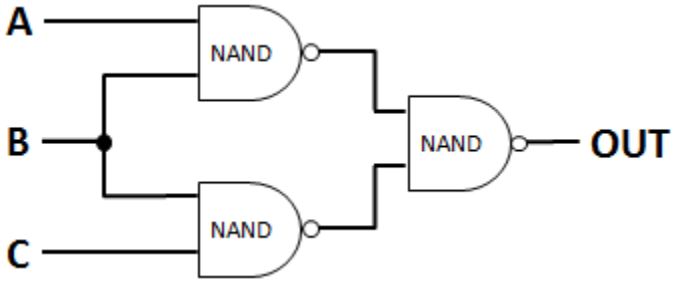
_____ Translate a program into machine language.

_____ Write a program in JavaScript.

_____ The ALU performs simple arithmetic.

_____ The CPU uses the fetch/decode/execute cycle to execute instructions.

3. Consider the following circuit:



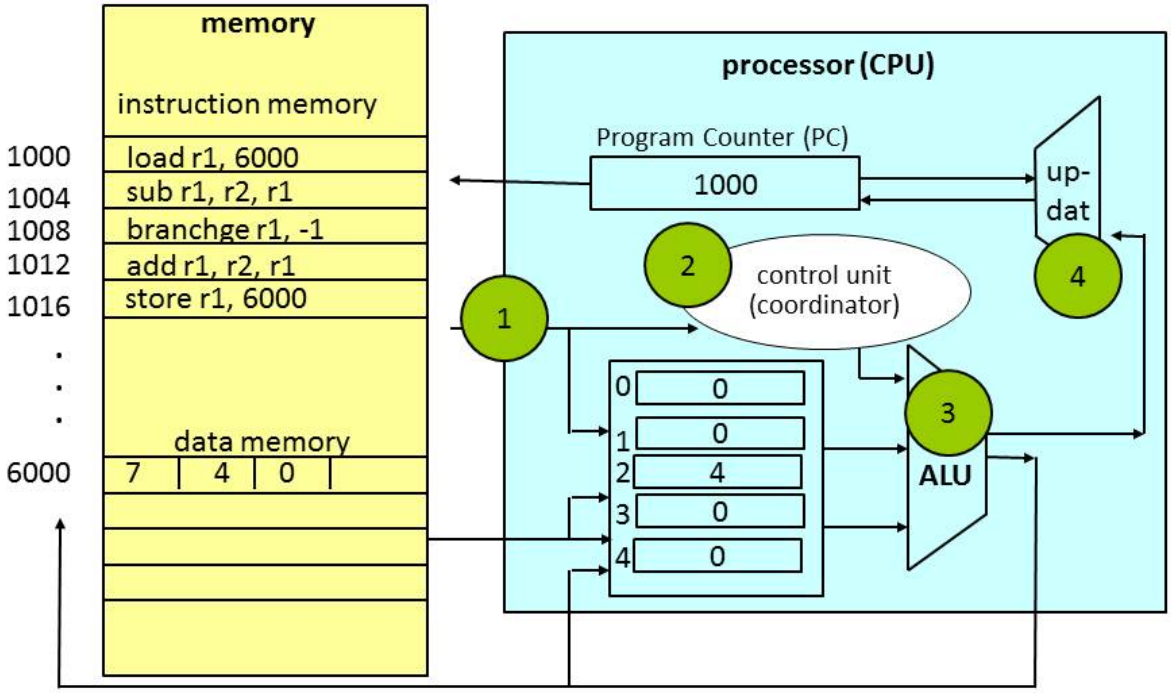
Complete the following table indicating how the circuit behaves: **[3 marks]**

A	B	C	OUT
F	F	F	F
F	F	T	F
F	T	F	F
F	T	T	T
T	F	F	
T	F	T	
T	T	F	
T	T	T	T

It may help to recall that a NAND gate with inputs X and Y functions according to the following table:

X	Y	NAND
F	F	T
F	T	T
T	F	T
T	T	F

4. Beginning with the diagram below, sketch the result of executing three instructions. Be sure to write in the values of data memory, the registers (numbered 0 through 4 in the diagram), and the program counter. Note: this question continues on the following pages. **[9 marks]**



Here is documentation on the five instructions used here:

`load r, mem:` load into register `r` the contents of memory location `mem`. For example, `load r0, 8000` would load the contents of memory location `8000` into register `0`.

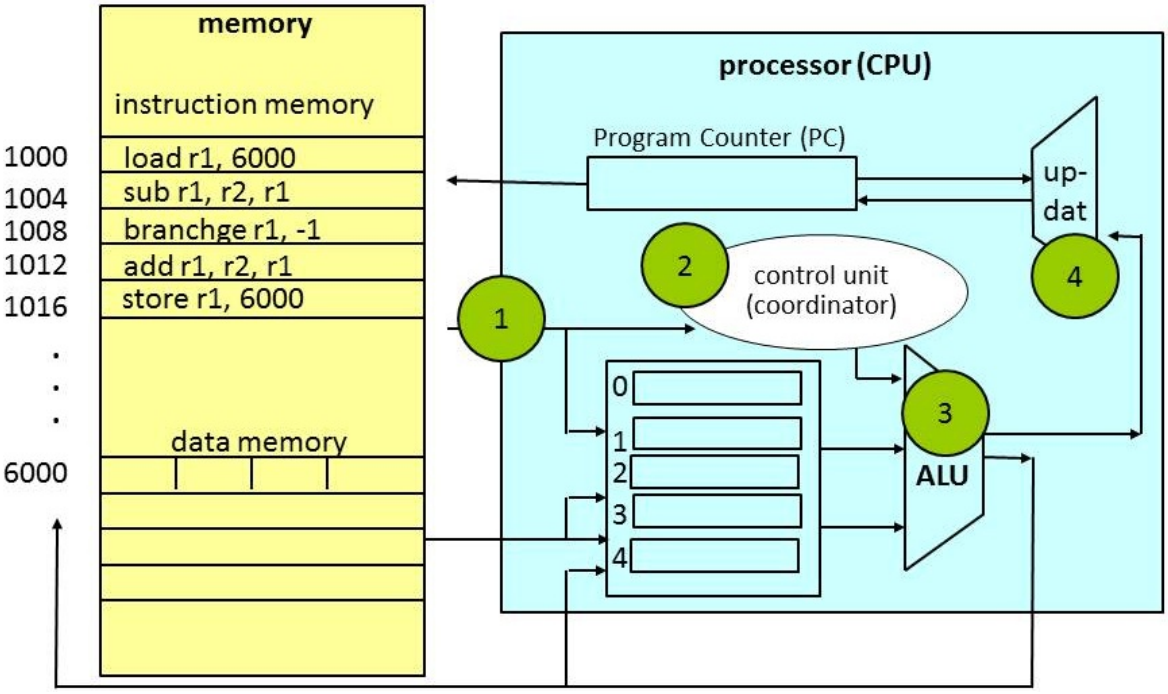
`store r, mem:` store the contents of register `r` into memory location `mem`. For example, `store r4, 8000` would store the contents of register `4` into memory location `8000`.

`add rA, rB, rC:` add the contents of register `rA` to the contents of register `rB` and put the result into register `rC`. For example, `add r4, r2, r1` would add the contents of register `4` to the contents of register `2` and store the result in register `1`.

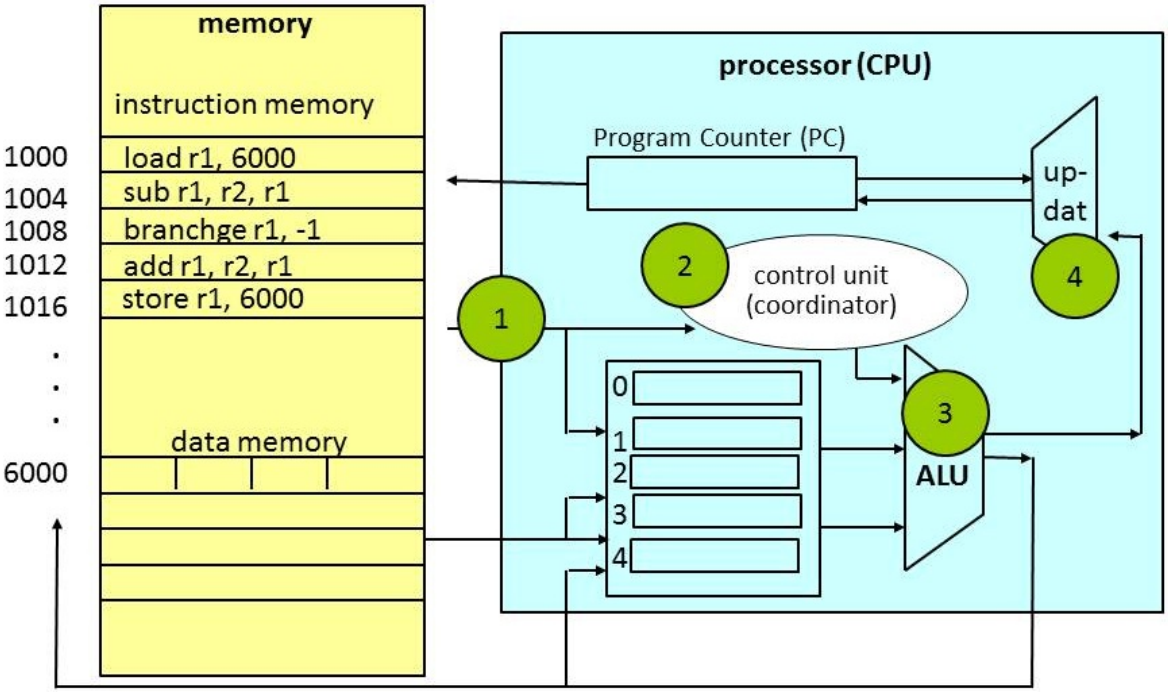
`sub rA, rB, rC:` subtract the contents of register `rB` from the contents of register `rA` and put the result into register `rC`. **Note that this is like the JavaScript statement: `rC = rB - rA`.** For example, `sub r4, r2, r1` would subtract the contents of register `2` from the contents of register `4` and store the result in register `1`.

`branchge r, num:` if the contents of register `r` are greater than or equal to zero, jump forward by `num` instructions. Otherwise, do nothing. For example, `branchge r5, -3` might do two different things. If the contents of register `5` were zero or positive, it would jump 3 instructions **backward** from the current instruction. If the contents of register `5` were negative, it would do nothing (and proceed as normal to the next instruction).

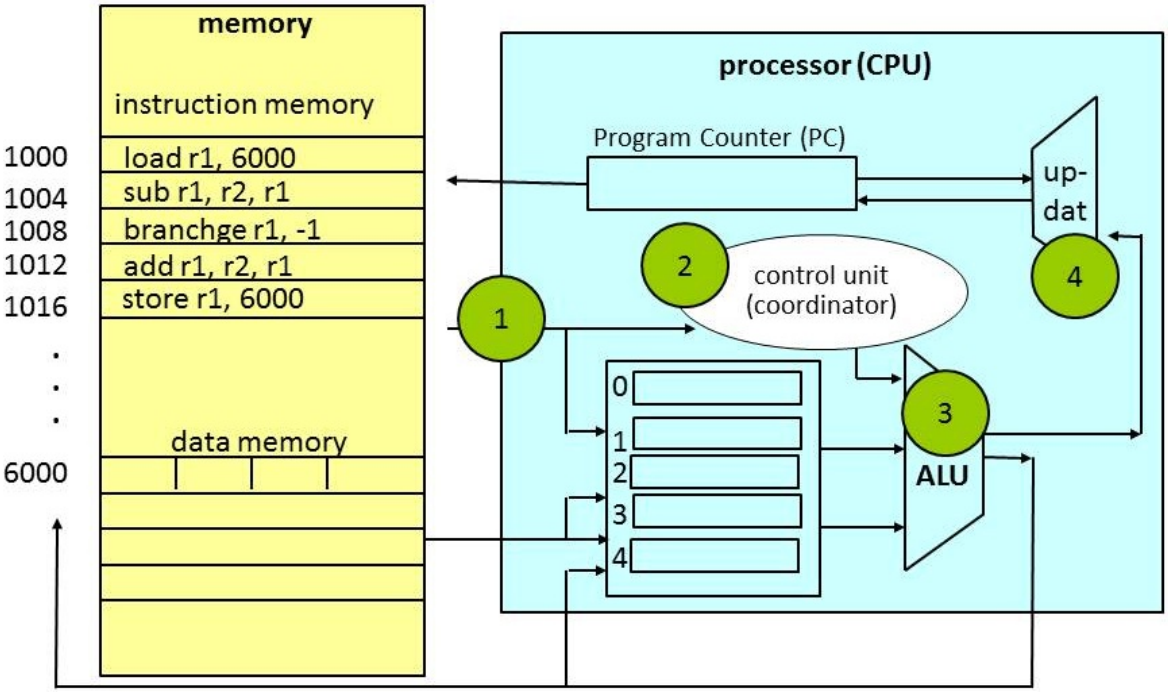
AFTER the FIRST instruction has finished executing and updated the program counter. (Fill in all of: data memory, the registers, and the program counter.)



AFTER the SECOND instruction has finished executing and updated the program counter. (Fill in all of: data memory, the registers, and the program counter.)



AFTER the THIRD instruction has finished executing and updated the program counter. (Fill in all of: data memory, the registers, and the program counter.)



5 Loop-de-loop [12 marks]

For each of the following, indicate as precisely as possible how often the code calls the `track()` function.

1.
```
for (var i = 0; i < 10; i++) {
  track();
}
```

2.
```
for (var i = 1; i < length; i++) {
  track();
}
```

3.
```
var i = 13;
while (i >= 10) {
  track();
  i = i - 1;
}
```

4.
```
for (var i = 0; i < 5; i++) {
  for (var j = 0; j < 3; j++) {
    track();
  }
}
```

5.
```
for (var i = 0; i < 10; i++) {
  if (i % 2 == 0) {
    track();
  }
}
```

6.
```
function foo(a, b) {
  var q = a + a;
  b = q;
  if (b <= a) {
    track();
  }
  else {
    track(); track();
  }
}

foo(5, 10);
foo(0, 3);
foo(1, -3);
```

6 JavaScript, HTML, and Algorithms [16 marks]

Consider the following web page code:

```
1.  <html><head>
2.  <title>Easily Annoyed Site</title>
3.  <script language="javascript">
4.  var annoyance = 0; // The number of times the user has annoyed the site.
5.
6.  // Respond when the user does something annoying.
7.  function annoy() {
8.    annoyance = annoyance + 1;
9.
10.   // Give a message appropriate to the annoyance level.
11.   if (annoyance < 2) {
12.     alert("Ow.");
13.   }
14.   else if (annoyance < 5) {
15.     alert("I'm giving you one last chance.");
16.   }
17.   else {
18.     // A while loop's body will keep executing until its condition
19.     // is false. Will the condition (0 == 0) ever be false?
20.     while (0 == 0) {
21.       alert("I warned you!");
22.     }
23.   }
24. }
25. </script>
26. </head>
27.
28. <body bgcolor="#F01010">
29. <h1>The Easily Annoyed Site</h1>
30.
31. <font color="red">Please be gentle with this site!</font>
32. Don't bother the frowny face!
33.
34. <p>
35.
36. <!-- An image that gets annoyed when you move your mouse over it. -->
37. <img src="frowny-face.gif" onmouseover="annoy()">
38.
39. <p>
40.
41. <em>Especially</em>, do not:
42. <input type="button" value="Click Here" onclick="annoy(); annoy()">
43.
44. </body></html>
```

Here is the image `frowny-face.gif`: 

1. Ignoring colour, sketch what the page looks like when it loads. **[4 marks]**

2. For each of the following programming concepts, give a line number or range of line numbers from the code with an example of that concept. **[3 marks]**

- an HTML comment:
- a loop:
- a JavaScript comment:

3. There is a severe HCI problem with the display of the text "Please be gentle with this site!" on this webpage. Explain what it is. (Hint: don't overlook the `<body ... >` tag!) **[3 marks]**

4. What are the two things the user can do that will cause the `annoy` function to be called? **[3 marks]**

5. What message will the user see the third time the `annoy` function is called? **[3 marks]**

7 Networks [9 marks]

1. IP addresses are currently represented using 32 bits, but many people want to change them to 128 bits. Why we might want more than 32 bits for IP addresses? **[3 marks]**

2. Recall the "party protocol" for sending information over a broadcast network:

```
1.   repeat
2.      listen to channel
3.      if channel is busy
4.      then wait for a fixed number of time steps
5.      else
6.              send packet on channel
7.              listen for packet on channel
8.              if packet is detected on channel
9.              then you are done!
10.             else wait a random number of time steps
11. until you are done
```

What might go wrong if we eliminated lines 2 through 5 from the algorithm above? (Hint: imagine I am busy sending a packet when you run the algorithm.) **[3 marks]**

If every computer has a unique IP address, we could avoid using randomness by telling every computer to wait as many milliseconds (thousandths of a second) as its IP address in step 10. Why might this be unfair? **[3 marks]**

8 Missing Link [3 marks]

Remember the following JavaScript code that prints a link to the UBC website into the current page?

```
<script language="javascript">
document.write("<a href=http://www.ubc.ca/>http://www.ubc.ca/</a>");
</script>
```

Below, we've begun to alter the code so that it uses a variable. **Finish altering the code to use a variable**:

```
<script language="javascript">



document.write("<a href=" + url + ">" + url + "</a>");
</script>
```

This page intentionally left (almost) blank.
If you put solutions here or anywhere other than the blank provided for each solution, you must *clearly* indicate which problem the solution goes with and also indicate where the solution is at the designated area for that problem's solution.

APPENDIX: HTML and UNIX

Summary of HTML tags:

Start Tag	End Tag	Meaning
`<html>`	`</html>`	surrounds whole HTML document
`<head>`	`</head>`	preliminary material at start of the page
`<title>`	`</title>`	title bar text; describes page
`<body>`	`</body>`	the main part of the page, change text color with text attribute
`<p>`	`</p>`	paragraph
`<hr>`		line (horizontal rule), can use width and size attributes
`<h1>...<h8>`	`</h1>...</h8>`	headings, eight levels, use in order, can use align attribute
`<font color="...">`	`</font>`	change the font color
`<strong>`	`</strong>`	strong emphasis (usually bold)
`<em>`	`</em>`	emphasise (usually italics)
`<a href="...">`	`</a>`	anchor reference
`<img src="...">`		image source reference (can use `onmouseover` for behaviour when mouse moves over image)
`<table>`	`</table>`	table, can use border attribute
`<th>`	`</th>`	table heading
`<td>`	`</td>`	table data
`<!--`	`-->`	HTML comment (not really a tag); ignored by the browser
`<input type="...">`		input field; the type must be an input type like `button` or `text`; can use properties like `value` (for button text) and `onclick` (for behaviour when clicked)

Note also that we enclose JavaScript code with `<script language="javascript">...</script>`.

Summary of UNIX commands:

Command	Meaning
`ls`	list contents of the current directory
`cd dir`	move to the "child" directory `dir`
`cd ..`	move "up" to the parent directory
`more file1 ...`	display file contents
`sort file1 ...`	sort file lines and display
`sort -u file1 ...`	sort, suppressing duplicate lines
`cp file1 file2`	copy `file1` to `file2`
`cp file1 ... filek dir`	copy files into directory `dir`

Summary of UNIX special characters:

Character	Meaning
`|`	connect the output of the previous command to the input of the next
`>`	connect the output of the previous command to the file listed after >
`<`	connect the input of the previous command to the file listed after <
`/`	separates directories in a path
`~`	the current user's home directory
`.`	the current directory in a path
`..`	the parent directory in a path
`*`	matches any number of characters in a filename (including none)
`?`	matches any one character in a filename
`\`	the next character is just what it seems to be, **not** special

APPENDIX: Powers of 2 and Hexadecimal Digits

Table of Powers of 2:

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8
1	2	4	8	16	32	64	128	256

Also, 2^{10} is about $1,000$ (a thousand), 2^{20} is about $1,000,000$ (a million), and 2^{30} is about $1,000,000,000$ (a billion).

Table of Hexadecimal Digits:

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111