Test Review for Part One

1. A Lab Exam and a Written Exam

The lab exam will test your ability to do the basic operations we have been practicing in lab and on homework. You will login, write a short scratch script, then login to Unix, work with some files and directories, use pico to create and edit a simple text file, then write a short web page with a table, a few images, a few links. You might even have to write a short shell script that uses a loop and a variable.

The written exam will ask you to define terms, explain facts, read code, modify some code, even write a short script.

The material on the exams will be based on class handouts, lab projects, and homework exercises.

2. Information, Instructions, Transmission

This review sheet provides an overview of the main topics in part 1 of Introduction to Digital Information. This review sheet also includes several review exercises. These exercises provide good practice with the ideas and skills of this part of the course. They do not represent typical test questions, but parts of them do.

This list is not comprehensive. Items, terms, facts, or skills not included in this review may appear on the exam. Consider this list a place to start. Read over your notes, exercises, and labs to fill in more details.

3. Quick Tip: Review Class and Lab Handouts

Every class has an outline. Make sure you can define each term, explain each idea, and perform each skill.

4. Most of All: Do Exercises

Using computers is an activity, not a spectator sport. The only way you gain comfort and fluency with a language, spoken or computer-oriented, is through speaking and using the language. Using a computer language means writing scripts or typing commands that get the computer to do things you want them to.

On the other hand, part of the course looks at ideas, terms, vocabulary, and skills. You have to know terms and facts to use the language meaningfully.

5. Six Main Topics

In the first five and a half weeks of the course, we have looked at six main topics. All these topics fall into the big three ideas of the course: algorithms to process data, storing data, presenting data.

The six main topics are:

- Algorithms: sequences of well-defined steps machines can perform
- Scratch and Programming Principles
- Using Unix to organize files/data
- Using convert to process image files
- Using HTML to control web-page appearance
- Scripting Unix to automate data/image processing

6. Algorithms: Sequences of Well-Defined Steps Machines Can Perform

Computer science is a field that solves problems by devising, studying, and applying algorithms. An algorithm is a sequence of well-defined steps a machine can perform. Devising a computer program requires two skills: creating the algorithm and expressing the algorithm in a language a computer can follow.

Getting a computer to solve a problem is similar to getting a music machine to play a song. First, you have to figure out the sequence of notes, and second, you have to express those notes in a language a machine can read. We saw some movies of machines that play music.
Holes punched in a sequence of cards control airflow in a 19th Century musicbox. Each hole is an instruction telling the machine to play a certain horn, ring a bell, tap a drum. From a different perspective, one can see these holes as information, a way to represent music and action.

Once one has devised a coding system, one can translate music, words, action into that system. How can one then transmit that encoding to someone else so that person can hear the music or read the words?

Music boxes introduce a few other crucial facts about computers: their limitations. The music boxes we saw can do specific actions. The first one we saw has organ pipes, bells, and a drum or two. The machine can toot the pipe, ring the bells, and hit the drum. The holes in the program cards can produced those actions in any order, in any combination, at any time. And that is all it can do. If you want to tell the music box to play a movie recorded on a DVD, you are out of luck; that is not something the machine can do.

Modern computers also do specific actions. They can store numbers in their memory and retrieve those numbers, store sequences of instructions, perform arithmetic operations, repeat sequences of instructions, compute Boolean expressions, execute conditional statements, read data in from the outside world and send data out to the outside world. They cannot do anything else. They can perform these limited operations in any order, many combinations, at any time. Any actions that can be represented as a sequence of these basic operations is doable by a computer. Any action that cannot be represented by a sequence of these basic operations cannot be done by a computer.

6.1. Ideas/Terms/Facts
You should be able to explain and/or define the following terms, facts, ideas.

- instruction
- information
- notation
- automated music machine
- programmable machines
- digital image
- pixel
- list of numbers
- operations on a list of numbers

6.2. Skills
We began the term by looking at two different problems: how to sort a stack of cards, and how to transmit a message to a collection of people. These are some skills you should know:

1. Compare systems of data transmission - speed, reliability, simplicity
2. Finding the lowest/greatest card in a set
3. Sorting a sequence of cards
4. Compare two systems for processing a list of cards

6.3. Questions
These questions may or may not appear on a test. Writing out answers to these questions is good preparation for the test.

1. How does the music machine sequence of cards differ from sheet music?
2. How is the music machine sequence of cards similar to sheet music?
3. What is a programmable machine?
4. What form do programs take?
5. What do the principles of laziness, impatience, and hubris require?
7. **Scratch and Programming Principles**

After looking at a few systems for encoding and processing information (music boxes, digital images) we shift our attention to a modern system for encoding actions and processing information -- Scratch. The Scratch programming language is a system for encoding actions. The actions Scratch encodes control 'sprites' on a 'stage' the same way the script of a play encodes instructions for actors and lighting and sound technicians.

Scratch introduces many of the basics of computer languages.

7.1. **Ideas/Terms/Facts**

Make sure you can explain and/or give examples of each of these terms/ideas:

- statement
- Boolean expression
- numeric expression
- Boolean operations (and, or, not)
- compound Boolean expression
- numeric operations (+, -, *, /)
- conditional execution (if..then..else)
- loops: forever, fixed, conditional counter
- variable
- nested loop
- nested conditional
- variables: set x to (), change x by ()
- variables: global/local, public/private (not fall 08)
- motion: relative, absolute, (dependent not fall 08)
- variables for counting
- variables for communication
- variables for running total (accumulator)
- variables for flexibility
- properties of sprites (x, y, size, color, costume)
- properties of stage (mouse_x, mouse_y, timer, loudness)
- drawing using the pen
- events: mouse click, keypress, touching

You can use variables in place of values in any statement.
You can use variables in Boolean tests to make decisions.
You can change the value stored in a variable to change sprite behavior.

7.2. **Skills**

Scratch is a big system. Programmers need many skills to use Scratch well. Here are some skills:

- a. Create a simple scratch project
- b. Create one or more scripts for a sprite
- c. Creating a new sprite
- d. Creating a new costume for a sprite
- e. setting the value of a variable
- f. changing the value of a variable by a fixed amount
- g. changing the value of a variable by a fraction of the current amount
- h. Using a variable to add up several values
- i. Using a variable to count events
- j. Using the sprite pen to draw patterns
- k. Using a conditional expression in an "if..then..else" piece
1. Using a conditional expression in a "repeat until <>" piece
m. Writing code that relies on the position and size of a sprite
n. Writing code that relies on the mouse position and/or the timer
o. Creating a variable
p. Incrementing a variable
q. Using a variable in an if.then.. statement
r. Make a loop repeat until a variable reaches a value

7.3. Sample Problems

1. Consider a sprite with two variables, x and y. At the start of the script, x holds the value 5, and y holds the value 7. For each of these expressions, state the type of the expression (Boolean or numeric), and state the value of the expression:

<table>
<thead>
<tr>
<th>expression</th>
<th>type</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 + 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x &lt; y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x &lt; 10 AND y &gt; 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>not ( x = 2 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x + 2 - y * 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-(x + 2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x = 2 OR x = 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x = 2 AND x = 5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. Write a one statement equivalent to:
   goto x: (0) y: (0)
   change x by (10)
   change y by (15)

3. Modify this script so
   a. The sprite says "enough!" after the user presses 'm' five times:
   b. the script modifies the presses variable using change instead of set
   
   when flag pressed
   set presses to (0)

   when [m] key pressed
   set presses to ((presses)+(1))
   if presses >= 10
   then
   say "Enough!"
   else
   say "Meow"

4. Compare and contrast the roles and meanings of the two instructions
   play sound [meow] and repeat (10)
   
   Some instructions tell the computer what action to do (move, play, turn). Some instructions tell the computer how or when to do an action. For example, an if..then..else instruction does not specify an action. Instead, this instruction tell the computer when to do one instruction and when to do another. Similarly, a repeat instruction tells the computer when to do an operation or how many times to do that operation.

   Question: name 5 Scratch instructions that directly affect a sprite and name 5 Scratch instructions that tell the computer when or how do do an instruction.

5. Write a Scratch project that adds up the first 100 squares and says the result. That is, the program should add up 1 + 4 + 9 + 16 + ... + 10000 using a loop and two variables.
6. Write a script that causes a sprite to turn 15 degrees clockwise each time it is clicked.

7. Write a script that causes the sprite to move forward a random distance (between 1 and 10 steps) each second.

8. Write a script that causes a sprite to shift its (x,y) position by a random x amount and a random y amount each time the 'm' key is pressed.

9. Write a script that causes a sprite to keep moving by 10 units every tenth of a second and stops when the sprite has bounced off the edge ten times. Can you use a fixed repetition loop to do this? Why or why not?

10. Given that the variable x stores the value 10 and the variable y stores the value -3, what is the value of each of these expressions:
   a. 2 < 3
   b. x < y
   c. x = 10 AND y = 0
   d. x = 10 OR y = 0
   e. (x + y) < (2 * x)

11. In math, one can write "less than or equal to" using the notation \( x \leq y \). How would one translate the following into equivalent code in Scratch?

   ```
   if x \leq y
   then
   ```

11. At the end of this script, what does the sprite "Say"?

   ```
   set x to (0)
   set y to (1)
   repeat until x > 20
   set x to (x + y)
   change y by x
   say x
   ```

12. In this script, how many times does the sprite say "Hello"?

   ```
   set x to (1)
   repeat until (x > 10)
   if ((x) > 3 and (x) < 6) OR ((x) = (1))
   then
   say "Hello" for (1) secs
   else
   wait (1) secs
   say " " for (0.1) secs
   set x to (x + 2)
   ```

13. In this script, what pattern does the sprite draw?

   ```
   pen down
   point in direction (90 [right])
   repeat (3)
   repeat (4)
   move 10 steps
   turn cw 90 degrees
   move 20 steps
   ```

14. Write a Scratch script that causes a sprite to draw this S-shape path starting from the lower left point marked A.

   ```
   ```

15. Write a Scratch script that causes a sprite to draw this saw-tooth pattern with 10 teeth.
The angle at the lower left point, relative to the horizontal, is 60 degrees, and the interior angle of each saw tooth is 60 degrees.

16. Explain the differences between local and global variables in the context of Scratch. When would you need a global variable? When is a local variable appropriate? (not fall 08)

17. What is the difference between the Scratch statements:
   
   set v to (____) and change v by (___)
   
   Rewrite change v by (10) using the set v to (____) piece.

8. Using Unix to Organize Files/Data

A digital image encodes an image by storing the spots of color that make up the picture. Different file formats (bmp, gif, jpg, etc) store the data in different formats. All record, as best as they can, the set of dots of color that make up an image. Regardless of which format you use, you have to store these encoded versions of images on a website so you can show them to your friends and family.

Once you have lots of pictures stored in files, you need a storage and/or filing system. Where do you store all these files, and how do you store these files in the folders where they belong?

One system for organizing images or other files is to group the images into categories, sub-categories, sub-sub-categories, etc. For example, a collection of photographs could be organized into three main categories: school, home, other. In photographs in the 'school' category, we can sub-divide these into one group of images of people, and one group of images of places (like buildings, fields, city squares). In the category for people, we could have one group of images for friends, one for faculty, one for other people. Of course, there are lots of ways to organize a collection of photographs into categories and sub-categories. A filing system reflects the way you, the person doing the filing, thinks about the contents.

In the physical world, you might buy several photo albums and organize each album by location, time, people, or any other category. In the digital world, you can create a directory for each album and then create sub-directories within each directory. This multi-level system reflects the groupings, sub-groupings, and sub-sub-groupings you use to make order from hundreds or thousands of images.

But how do you make these groupings and sub-groupings, and how do you move images around through this collection of folders, sub-folders, etc?

Many web servers use the Unix operating system. Unix is a system designed to allow many different people to store and manage files on a single machine. Unlike Windows or the Mac OS, Unix was built at the outset to be used by several people at once and to run several programs at the same time. Getting used to using Unix requires a clear understanding of three main topics: users, directories files.

8.1. The Home Directory

On a single Unix computer, several people can be doing different things at once. Some people can be reading or sending email, several can be surfing the web, several can be writing computer programs, and others can be writing documents, running mathematical computations, almost anything.

To keep the work of each user separate from the work of other users, Unix uses the idea of a Home Directory of each user. The disk is divided into lots of folders; each user has a personal folder for regular, private use. When a user logs into the system, he or she starts work in his or her home directory. A user may create any number of subdirectories and sub-subdirectories below his or her home directory.

8.2. File Ownership and Access Permission

On a Unix multi-user machine, each file has an owner. Furthermore, each file has a group to which it belongs. For example, a file can belong to a user named jane and to a group called sales.
If user Jane wants this document to be private, she can set the file permissions to prevent members of the 'sales' group or other people from reading the file. Or, Jane may want to share a file with people in the 'sales' group but not allow these people to make any changes to the file.

Unix provides three types of people who might want to view a file or run a program, and for each of those three types, Unix provides three types of access. The three types of users are user, group, and other. The three types of access are read, write, and execute.

### 8.3. Ideas/Terms/Facts

You should be able to explain these terms. For each, you should know what the term means, why that concept is useful, and where that concept fits into the larger picture.

1. directory
2. files
3. home directory
4. website
5. pwd
6. directory operations: mkdir, rmdir, ls, cd, pwd, chmod
7. file operations: cp, rm, mv, cat, ls -1, pico -w
8. wildcards: *.jpg -- shorthand for identifying a bunch of files/dirs
9. ssh, fugu, remote file transfer
10. A website is a directory called public_html on a server
11. You put files on a website by putting them into public_html
12. You organize files on a website by making subdirectories below public_html
13. home directory
14. public_html
15. permission mode
16. parent directory
17. directories called . and..
18. using directory paths to refer to files and directories

### 8.4. Wild Card Summary

You may abbreviate list of file or directory names by using the wildcard shorthand. The wildcard notation includes three special symbols:

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MEANING</th>
<th>EXAMPLE</th>
</tr>
</thead>
</table>
| *      | matches any sequence | mv *.jpg small  
chmod o+r album1/captions/*  
for x in *.jpg *.JPG |
| ?      | matches any single character | cp cap???.txt captions/10-99 |
| [list] | matches one char in a list or range | mv cap[12345].txt notes/firstfive  
mv cap[1-5].txt notes/firstfive  
mv *.([Jj][Ff][Gg]) small  
for f in [Cc]ap[1-9][ab].gif |

### 8.5. Skills

To build a website, you have to decide how you are going to organize your images, documents, and/or tunes. Unix allows you to use directories and subdirectories to create a filing system. Using this system effectively requires that you know how to work with directories, files, how to move them, copy them, rename them, and give make these files and directories accessible to others. In particular, you need to know the skills of:

1. Create a directory
2. Explain the difference between home directory and website
3. What permission you need for your home directory
4. How to set access permission for a file or directory
5. How to rename a file or directory
6. How to move files from one directory to another
7. Draw a picture of a directory tree by exploring it
8. Draw a picture of a directory tree by reading commands to build it
9. Use wildcards to issue commands shortly and effectively.
10. Use ssh to connect to a remote server
11. Use pico to create and modify files on the server
12. Create a website on the server
13. How to transfer files to and from the Unix server

9. Questions

Using the skills mentioned in the previous section requires a lot of practice. These questions ask you to practice basic skills of building a website and installing readable data.

1. A website is a subdirectory of your home directory. What is the name of the subdirectory that will be your website, and what access permission does it need to have?

   For the following few questions, please refer to this diagram:

   ![Diagram]

2. What sequence of UNIX commands would create the tree shown in the diagram? Assume that the directory $T$ is directly below $HOME/public_html$

3. The diagram shows a file called $f1$ in directory $T/A$ and a file with the same name in $T/B/E$. Say you wanted to place a copy of file $f1$ in all five directories that do not yet contain a copy of $f1$. Write down a sequence of commands that would place a copy of the file $f1$ in $T/A$ in all five empty directories in this diagram. Assume you begin in directory $T$ in this diagram.

4. If you are currently in the directory marked $D$ in the diagram, what one-line command will transport you to the directory marked $F$ on the diagram?

5. The diagram does not show what files are stored in the directory called $F$. Write the commands that will, assuming you are in directory $F$,
   a. Create subdirectories called $P$, $Q$, $R$, $S$, $T$ one level below the $F$ directory below $C$, below $T$, below your home directory.
   b. Copy all the files currently stored in the directory called $C$ in the diagram into each of the new directories under $F$.

6. What is the connection between your home directory and your website?

7. What access permission settings do you need to apply to a directory on a website?

8. What access permission settings do you need to apply to a plain text or html file on a website?

9. What access permissions do you need to set for a shell script you do not want executed by other people or by other members of your group?

10. Most of our examples focus on storing digital images. How can you distribute your digital pictures to friends and families? The simplest way is to store the images on a server and then tell your visitors the web address where you have stored your pictures. With the storage capacity of digital
cameras increasing all the time and server space getting ever more plentiful, you can store a lot of images on a server.

How do you make order out of all your digital images? The simple answer is to group the images into topics and subtopics. Another solution is to group the images in chronological order.

Devise a filing system for your images that uses two or more levels to put the images in categories and subcategories. Draw your filing system as a tree diagram.

11. If you were in the directory called E and wanted to move the file called f1 into the directory called D, what command could you type to relocate the file?

12. The Unix mv command can rename or relocate a file or directory. Give an example where mv will rename a file, and give an example where mv will relocate a file.

13. What is the effect of these two commands:

   mkdir dirB
   mv *.JPG dirB

10. Using convert to Process Image Files

   Modern digital cameras take pictures at high resolution. A single picture may consume over a million bytes of disk storage to record on image. One advantage of this high resolution is that the images show a lot of detail. Some disadvantages of high resolution include the inconvenience of downloading large files, particularly for viewers with slow connections and the inconvenience of seeing much more detail than most visitors want.

   A simple solution is to create small versions of each large picture. We can display these small images to our visitors. If someone wants to see detailed, larger version, the visitor can click on the small image to retrieve the larger version.

   But how do we create small versions of files? There are several programs available that will process image files. A popular, and successful one is "ImageMagick" and its convert tool. This convert program reads in an original image and writes out a modified version of the original. For our needs, a smaller version is just what we need. For other applications, though, we might need other conversions between original image and result.

10.1. Ideas/Terms/Facts

   convert is part of the ImageMagick toolset. This toolset includes programs to operate on image files in many ways. You should be familiar with the following facts/ideas/terms:

   1. image size in pixels
   2. original image, output image
   3. the -resize operation for convert
   4. other operations for convert

10.2. Skills

   Using convert is pretty easy. Convert is a command-line program. That is, you type to Unix the command:

   convert -resize 120x120 mypet.jpg pet-small.jpg

   this command runs the convert program and tells the convert program three things:
   a. The original file is mypet.jpg
   b. The output file is pet-small.jpg
   c. The processing should be -resize 120x120

   Convert can operate in some other manners, but the simplest version for our purposes is that convert reads an input file, does some processing, then writes an output result.
10.3. Skills, Continued

Convert can transform images in many ways. For our needs of building an album or slideshow, we need to look at scaling images first. Later, we might explore how to work with smaller images.

11. Shell Scripts

In the same way that Scratch allows us to click on individual puzzle pieces to execute individual instructions and also to combine pieces into scripts, Unix allows us to type single commands and also to combine commands into scripts. And just as Scratch provides loops, conditionals, and Boolean expressions to control when, how, and if certain blocks of statements are executed, the Unix shell provides loops, conditionals, and Boolean expressions that control when, how, and if certain blocks of commands are executed.

11.1. Shell Script Ideas/Terms/Facts

Scripts are plain text files created with an editor like pico. Here are some facts:

1. A script is a text file that contains a list of commands
2. A script starts with the magic phrase: 
3. Make a script executable with
4. A script can contain comments - these are lines that start with #
5. A script can contain loops
6. A `for` loop in a shell script has the format
7. The items in the list are separated by spaces
8. The list of words may contain wildcards to represent several files at once
9. To refer to the value of a variable, precede its name with $
10. The special symbols $1, $2, $3, ... are place holders for words on
the command line to the script
11. The symbols $1, $2, $3, .. are called parameters of the script
12. The values given on the command line are called arguments
13. The shell script copies the arguments into their placeholders
14. A script can call other scripts
15. A script can even call itself
16. A script that calls itself is called a recursive script

11.2. Shell Script Skills

The basic skills of using scripts involve creating them, using loops, variables, wildcards, and conditional execution.

1. How to login to linux.eecs.tufts.edu
2. How to use chmod to set attributes for files and directories
3. How to use pico to create and/or edit a script
4. How to use a for loop in a shell script
5. How to use wildcards to refer to groups of files or directories
6. How to use while in shell scripts
7. How to use counter variables in shell scripts (see lab05)

11.3. Shell Script Questions

1. What does this script do?
2. Use one or more loops to rewrite this script so it has the same effect but is shorter:

   cd public_html/pictures
   mkdir travel
   cd travel
   mkdir 2007
   cd 2007
   mkdir places people
   mkdir places/small people/small
   cd ..
   mkdir 2006
   cd 2006
   mkdir places people
   mkdir places/small people/small
   cd ..
   mkdir 2005
   cd 2005
   mkdir places people
   mkdir places/small people/small
   cd ..
   mkdir 2004
   cd 2004
   mkdir places people
   mkdir places/small people/small
   cd ..

3. Given this script called mystery:

   #!/bin/sh
   # a script that takes two arguments
   #
   for pic in 1.jpg 2.jpg 3.jpg
   do
     for size in 100x100 200x200 300x300
     do
       convert -resize $size -charcoal $1 $pic effects/$2/$pic
     done
   done

   What is the effect of these two commands:

   ./mystery 3 drawings

   for x in 3 5 8
   do
     ./mystery $x drawings$x
   done

4. What is the difference between `ls` and `ls -l`?

5. What is the difference between `cp` and `mv`?

6. What does the command `who` do?

7. Write a shell script that uses a loop to print out:

   <table border='1' cellpadding='2' cellspacing='0'>
   <tr>
     <td><img src='small/1.jpg'></td>
     <td>Click <a href='big/1.jpg'>here</a> for picture 1</td>
   </tr>
   <tr>
     <td><img src='small/2.jpg'></td>
     <td>Click <a href='big/2.jpg'>here</a> for picture 2</td>
   </tr>
</table>
8. Write a shell script that uses a nested loop to print out:

```
Our breakfast choices today are:

spam and eggs
spam and toast
spam and fries
spam and muffin

bacon and eggs
bacon and toast
bacon and fries
bacon and muffin

tofu and eggs
tofu and toast
tofu and fries
tofu and muffin
```

12. Material in Textbook

You are responsible for the material chapters 1, 2, 6, and 8 of the textbook. Go over the summaries at the end of the chapters and do the review exercises to practice.

13. HTML

The last topic for this part of the course is HTML. You need to know the purpose and use of the basic HTML tags. Tags play two roles in HTML. Some tags create containers that apply attributes to text. Other tags stand for items that do something or create something.

13.1. Web Pages Facts

1. A web page is a text file containing text and format codes
2. A web page can specify images to display
3. A web page can contain links -- items you click to see something
4. The convert program processes image files
5. A web page consists of a collection of elements
6. Each element on a web page can have several attributes
   (e.g. `<table border='0' cellpadding='2' cellspacing='3'>`)
7. The attributes of an element are specified in the element's tag
8. Different types of elements have different attributes
9. Any element can have a `style` attribute
   The style attribute consists of one or more style properties
   (e.g. `<p style='text-align:center; background-color:blue;' />`)

13.2. Web Pages Words

1. home directory
2. website
3. link
4. public_html
5. web page
6. image source
7. HTML
8. HTML tags
9. IMG tag
10. A HREF= tag
11. element
12. tag attributes
13. lists in HTML
14. Tables in HTML
15. Style properties
16. Tag attributes
17. Style sheet file
18. element attributes
19. Style sheet file
20. table
21. table row, table data

13.3. Web Pages Skills

1. How to use pico to write a simple HTML page
2. How to create a list in HTML
3. How to make a text link to a web page
4. How to add an image to a web page
5. How to make an image a link on a web page
6. How to set element attributes to control the look of a page
7. How to use tables in HTML
8. How to make a cell span more than one column
9. How to use shell variables to automate the convert command

13.4. Some Random Facts and An Exercise

The tag <HR /> creates something, a horizontal rule. The tag <BR /> does something, inserts a line break in the text.

You should be able to use tables, make links to images and links to web pages, insert images, use images as links.

Exercise

Write the HTML code to produce a table that looks like:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>b</td>
</tr>
<tr>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>x</td>
</tr>
</tbody>
</table>