CPSC 101: Connecting with Computer Science
Midterm Exam, 2012 February 7

SAMPLE SOLUTION

1 Cooking Up a Computer Question [5 marks]

Which part of our course definition of a computer makes clear that a human chef cooking a dish from a typical recipe is *not* a computer? Circle the letter before the **best** choice—a line from our definition—or indicate that the chef *is* a computer by this definition. Then, **briefly justify your answer**.

a. A system that receives a list of instructions...

b. ...drawn from a well-defined set of possible instructions...

c. ...and interprets them...

d. ...to perform some process in the world, such as physical activity or transformation of information.

e. None of these; the chef is a computer by this definition.

Briefly justify your answer:

Solution : We actually altered the definition slightly to make the answer "drawn from a well-defined set of possible instructions" more clearly the best. Many instructions in many recipes are fairly standard, but there's no comprehensive list or definition of the possible instructions a recipe can have, and the particular meaning of instructions is also just as vague and ambiguous as most English. (Our change was to alter "A device that receives..." to "A system that receives...".)

2 Missing Link [8 marks]

Consider 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>
```

1. Below, we've begun to alter the code so that it uses a variable. **Finish altering the code to use a variable**: **[4 marks]**

Solution :

```
<script language="javascript">

var url = "http://www.ubc.ca";


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

Note: there should really be quotes around the value of the `href` attribute in an `<a>` tag. We didn't include them because it looks messier. If we do include them, the last line of this code becomes:

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

Do you recognize the \ before the "? It's much like the \ mentioned in the appendix for UNIX. It means that the next character is just what it says it is: a quotation mark in the text, *not* the quotation mark that ends this bit of text.

2. Finally, why might we want to add this to the <head> part of our HTML document: **[4 marks]**

```
<script language="javascript">
function displayURL(url) {
  document.write("<a href=" + url + ">" + url + "</a>");
}
</script>
```

Solution : There's at least three natural reasons that we discussed in class. One is to be able to use this code over and over again easily. If we want to display several different URLs (or the same one several times), we can just call displayURL several times rather than writing out the longer, messier code. Another is that we can use this code to solve different problems: displaying any URL we want, not just the UBC website's URL. Finally, it helps us think at a higher level about problems we want to solve (like the postal system/network examples of abstraction where we get to ignore all the messy details of how a message gets from one place to another at the higher levels of abstraction).

3 Adventures in HTML [20 marks]

Consider the following four files:

Contents of the file ∼/web/index.html:

```
<html><head><title>Lobby</title></head>
<body>
<p>You are in the entrance.</p>

<p>You see a door to:

<!-- Show an unordered list of rooms that are reachable. -->

the <a href="rooms/parlor.html">Purple Parlor</a>

and the <a href="bye.html">outside</a>.

</p>
</body></html>
```

Contents of the file ∼/web/bye.html:

```
<html><head><title>Goodbye!</title></head>
<body>
<p>It was fun having you!</p>
</body></html>
```

Contents of the file ~/web/rooms/parlor.html:

```
<html><head><title>Parlor</title></head>
<body text="purple">
<p>Everything here is purple!</p>

<p>There's a door back to the <a href="../index.html">lobby</a> and
stairs leading to the <a href="xroads.html">crossroads</a>.</p>
</body></html>
```

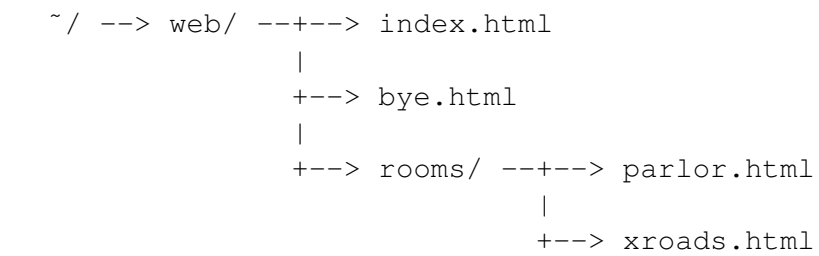
Contents of the file ~/web/rooms/xroads.html:

```
<html><head><title>Crossroads</title></head>
<body>
<p>A room called the crossroads, under a full moon.</p>

<p><!-- <a href="parlor.html">Stairs to the Parlor</a>. -->
The stairs collapsed!  Go through the <a href="../bye.html">window</a>!</p>
</body></html>
```

1. Draw a diagram of the file and folder structure as a hierarchy. **[4 marks]**

   Solution : We'll draw a crude text picture. We put a / after directories to make clear which of these are directories and which files (but you didn't need to!).



2. Draw a diagram of the structure of links in the website as a network. **[4 marks]**

   Solution : Continuing the crude pictures:

```
index.html  -----------> bye.html
  ^                        ^
  |                        |
  |                        |
  v                        |
parlor.html -----------> xroads.html
```

3. Give the result of the following UNIX commands: **[4 marks]**

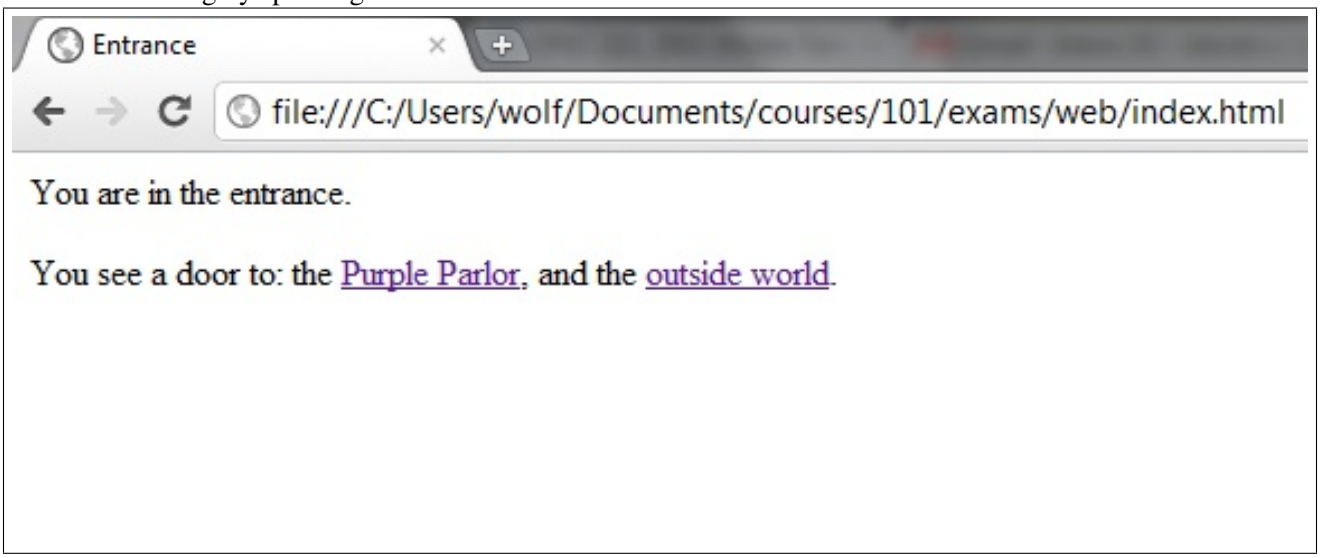
```
cd ~/web/rooms
ls *d*
```

Results:

Solution : We'll see just `xroads.html` listed, as it's the only one in the `~/web/rooms` directory that contains a `d` in the file's name.

4. Sketch what the file `~/web/index.html` would look like in a web browser: **[4 marks]**

Solution : Roughly speaking:



(The URL would probably look similar but not the same. Oops! Our testing code still had "Entrance" as the title, not "Lobby". Can you see where that changed the page? Our testing code also had a comma after the end of the `</a>` tag after the word "Parlor".)

5. Now, correct the HTML code (not the drawing above) for the page `~/web/index.html` so that it will have the author's intended appearance. **[4 marks]**

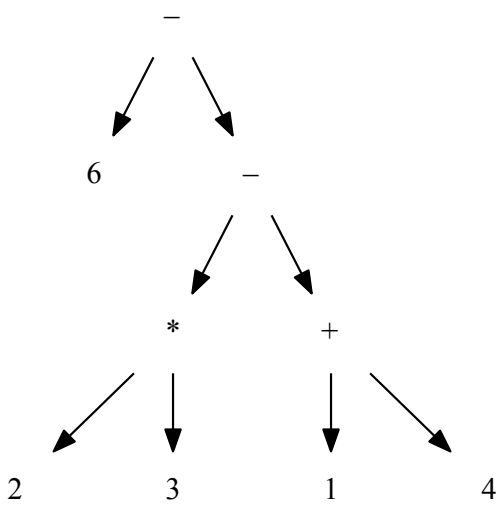
Solution : We'll just point out the necessary changes here. The comment suggests that the "places to go" should be in an unordered list. Remember that a web browser doesn't need to keep the spaces and line endings

("whitespace") you put into your HTML source; so, the fact that they're on separate lines doesn't matter. Instead, we'll use tags to express that to the web browser.

We wrap the two "places to go" in one `<ul>...</ul>` pair. Then, wrap each individual place to go in `<li>...</li>`.

4 Pyramid Scheme [10 marks]

Consider the following tree, which represents an arithmetic expression—including subtraction, multiplication, and addition—as a hierarchical structure. Convert the tree into a textual representation (using parentheses). Then, calculate the value of the expression, showing your calculation steps on the tree.



Textual representation: [5 marks]

Solution : Imagine putting parentheses "around" each item under a node and then "moving down" the operator (the $-$, $+$, or $*$) in between the items in those nodes:

$(6 - ((2 * 3) - (1 + 4)))$

Following the usual rules of arithmetic, it would also be OK to use:

$6 - (2 * 3 - (1 + 4))$

Calculated value of the expression: [5 marks]

Solution : Starting from the bottom "row" of operators, the $*$ gets the values 2 and 3 and multiplies them together to get 6. To its right, the $+$ adds 1 and 4 to get 5. Moving up, the lower $-$ gets the 6 from the $*$ and the 5 from the $+$ and calculates $6 - 5 = 1$. Finally, the top $-$ subtracts that 1 from 6 to get the final answer: 5.

Don't forget to write your calculation steps on the tree above!

5 . . . by Any Other Name [12 marks]

Names are crucial in computing and behave somewhat differently in different contexts. Answer each of the following questions about *incorrect* or *unusual* use of a name in a computing context. (**[2 marks]** per question.)

1. We change the name of a text file to `doc.pdf` and then double-click on the file. What's the most likely result?

   a. Nothing will happen.
   b. The usual thing will happen: Our text editor will open the file.
   c. Some application (other than our text editor) will open the file normally.
   d. Some application (other than our text editor) will attempt to open the file and fail.
   e. None of these is likely.

   Solution : The most likely answer is that some other application will attempt to open the file and fail, as we've changed the "type" of the file for our operating system. In particular, your PDF reader (likely Adobe Acrobat) will attempt to open the file and probably fail.

2. We write a piece of JavaScript code with the names mixed up:

```
var numberOfCS101Students = 4;
var classesPerStudent = 100;
alert("# of classes we take: " + (numberOfCS101Students * classesPerStudent));
```

   What's the most likely result?

   a. The program will give an error message because of the incorrect names.
   b. The program will not even start because of the incorrect names.
   c. The program will run and show a correct result in the `alert` window.
   d. The program will run but show an incorrect result in the `alert` window.
   e. None of these is likely.

   Solution : The names look reversed to a human. (There's about 100 students in `CS101` and each one takes about 4 classes on average, not the other way around.) But the computer will still happily multiply 4 by 100 and get the same answer it would have gotten multiplying 100 by 4. That doesn't make the *code* correct! It does make the result correct, however.

3. We attempt to open a URL like the following in a web browser: `http://142.103.6.43/~cs101/`. What is the most likely result? (Note: the IP address of `www.ugrad.cs.ubc.ca` is `142.103.6.43`.)

   a. The webpage will fail to load because this IP address is not unique to `www.ugrad.cs.ubc.ca`.
   b. The webpage will fail to load because only domain names can be used to identify computers on the Internet.
   c. The webpage will fail to load because humans do not understand IP addresses.
   d. The webpage will fail to load because the web browser does not understand IP addresses.

e. None of these is likely.

Solution : This would be a tough prediction to get absolutely correct beforehand. We believed from experience and knowledge that users occasionally want or need to enter IP addresses that "none of these" is the most likely result, and testing on several browsers shows that's correct. The web browser happily opens the webpage this way. (It simply skips DNS translation.)

However, process of elimination should definitely get rid of the first, second, and third answers. (The first is wrong or irrelevant, since that shouldn't happen but if it did would be just as much of a problem if we use the domain name rather than the IP address. The second is simply untrue. The third is true but not relevant because we're talking about the *web browser* understanding, not the human user.)

The fourth is a perfectly reasonable answer founded on the division between the abstraction layers.

4. Your computer sends out an e-mail message to `wolfman@cs.ubc.ca` rather than `wolf@cs.ubc.ca`. What is the most likely result?

   a. The e-mail goes through normally to `wolf@cs.ubc.ca`.
   b. The e-mail goes to the user `wolfman` at `cs.ubc.ca` if that user exists and otherwise fails to go through.
   c. The e-mail goes to the user `wolfman` at `cs.ubc.ca` if that user exists and otherwise goes through normally to `wolf@cs.ubc.ca`.
   d. The e-mail fails to go through no matter what.
   e. None of these is likely.

   Solution : Remember that to the computer these distinct names have no relationship to each other (just like in JavaScript). So, the answer is that the e-mail goes to the user `wolfman` at `cs.ubc.ca` if that user exists and otherwise fails to go through.

   By the way, the e-mail will almost certainly get as far as (the mail server computer for) `cs.ubc.ca`, since there's nothing wrong with the domain name.

5. You have a subdirectory of your home directory named `cs101dir` and you attempt to execute the following UNIX command in your home directory: `ccd cs101dir`. What is the most likely result?

   a. You change directories into `cs101dir`.
   b. You change directories into some other directory.
   c. The command fails, indicating that the directory `cs101dir` does not exist.
   d. The command fails, indicating that the command `ccd` does not exist.
   e. None of these is likely.

   Solution : Again, this is essentially the same question as the JavaScript and e-mail ones. There's no connection in the computer between `ccd` and `cd`. So, the command will fail indicating that `ccd` doesn't exist.

   Some very recent shells will actually suggest `cd` as the command after they discover that `ccd` doesn't exist, but they still indicate an error. For this reason, we'll also accept "none of these", although answer (d) still seems like a good fit in this case.

(Note that the shells will only do this if they "believe" based on configuration that they're working with a human, not with a program written using the shell commands. That's because they'd never want to make such a "correction" without a human's confirmation. The way they actually check this is with an algorithm that compares an input—but non-existent—command to a list of all the available commands based on a model of how humans make typos.

This sort of behaviour would be a very bad idea in JavaScript, because there's no human who understands JavaScript that's necessarily "in the loop" when the program runs. It would also be a bad idea in the e-mail case, for privacy reasons, if nothing else. There's no reason that the e-mail application you use to send the e-mail couldn't suggest another address to you, however, if it happens to "know about" `wolf@cs.ubc.ca`. That still doesn't solve the problem of when you do send to `wolfman@cs.ubc.ca`.)

6. You attempt to put some text on your webpage in bold with: `this is <bald>COOL</bald>`. (Check the appendix for the correct way to but text in bold in HTML!) What is the most likely result when you open the webpage in a web browser?

   a. The browser crashes. (Closes immediately with an error.)
   b. The browser does not display the page at all, giving an error message.
   c. The browser displays the page with the text: "this is COOL" (with COOL in normal type).
   d. The browser displays the page with the text: "this is **COOL**" (with **COOL** in bold).
   e. None of these is likely.

   Solution : By far the most likely answer is that the browser displays "this is COOL" with no bolding of the text. First, `bald` and the `b` of the bold tag are distinct names and so technically have no relationship to each other, as discussed previously.

   We've also discussed that web browsers try hard to display webpages with errors reasonably. As we saw in class with the `<stong>` typo, the browser just ignores an unfamiliar tag.

   However, it is plausible that a browser would try to get clever and do a *lot* of error correction to (1) get from `bald` to `bold` and then (2) "understand" that people might type `bold` meaning `b`. This is probably a bad idea, and it's not what we've seen, but we nonetheless gave full credit for the "this is **COOL**" in bold answer.

6 Playing Around [8 marks]

1. If you **don't know** what directory you're currently in and want to change to a particular new directory, which should you use, relative file paths or absolute file paths? **Briefly justify your answer. [3 marks]**

   Solution : Absolute. "Relative" file paths are relative *to the current directory*. If you don't know the current directory, then you can't predict the right relative file path to your target directory.

   (It's a perfectly good answer to say "either one", if you *clearly and briefly* explained that it is possible to determine the current directory with a command. You didn't need to name the command, but one that works is `pwd`.)

2. Assume you have the directories `~/play`, `~/play/test`, and `~/play/html`. Say you're currently in directory `~/play/test`. Give two ways (one using relative paths and one absolute) to change into directory `~/play/html`. **[5 marks]**

Solution : There are *many* answers. Here's two that will probably be the most common correct answers:

- `cd ../html`
- `cd ~/play/html`

7 Stringing Together a Network [11 marks]

We had four pieces of information on every "packet" we sent through our "network" when we did the point-to-point routing exercise at the front of the class: the recipient's address, the number of the current packet (where, for example, a 5 means "this is the fifth packet"), the total number of packets in the message, and a portion of the message itself. In this problem, we consider what might go wrong if we eliminate parts of this data from our agreed-upon protocol.

Solution : This wasn't the question, but it *is* interesting to ask what will happen if the protocol includes some information that we leave out. Let's use the "missing recipient" as an example.

In practice, an attempt to send such a "malformed" packet shouldn't get out of your computer unless you intentionally force it to with a low-level program. If it does get to an Internet router, one of two things will likely happen. To understand them, you need to know that the packet won't actually appear to be "missing" the receiver. A packet is really a (linear) sequence of bits. It's like filling out an application form with various blanks for name, etc., except that you *cannot* fill in a later blank until you've fully filled in an earlier one. (To make the analogy even better, there's also a certain fixed number of letters you have to put in each blank, as with a SIN but *not* with a human name. Because the computer has only 0 and 1 available and not an extra "blank character", there's no option to leave some of the letter slots "blank" unless the protocol agrees on a pattern of 0s and 1s that "means" blank.)

So, if you leave off the receiver, either you still need to choose some bits to put in for the receiver (like all 0s), or you move other bits forward.

In the former case, the router will think that the bits you put in as a "blank" receiver address really are the receiver address and interpret them that way. Either it won't be a valid address—in which case the packet doesn't send—or it will be—in which case the usual thing happens since you really did put a recipient on the packet.

In the latter case, the router will interpret whatever bits you brought forward as an IP address, probably getting a sort-of random IP address as a result. So, the Internet router will likely either drop the packet—possibly sending a packet back to your computer indicating that the recipient IP address was unknown or malformed—or it will send the packet along to whatever "autonomous system" (portion of the Internet) is responsible for IP addresses in the range the recipient appears to be in, and once it gets there, *that* portion of the network will either drop it (because the address really doesn't exist) or deliver it to the wrong computer if one with that IP address happens to exist. If it goes to the wrong computer, it will probably get dropped at that point. In the unlikely event that that computer is able to extract some meaning from the malformed packet and happens to be allowing messages from the network of whatever type the packet seems to be, that computer might actually act on the spurious information.

What if your packet loses or alters a bit while going over the network? Does the same sort of thing happen? The answer is very probably *no*. Remember when we looked at "parity bits" to check if data in the computer is valid? It's an extra bit that is 1 if the current block of data had an odd number of 1s in it and 0 otherwise. Then, you can count up the 1s including the parity bit, and you should always see an even number. If there's an odd number, you know an error occurred. Internet routers use something similar—called a checksum—to detect errors and drop packets where errors have crept in. Just like a parity bit, checksums aren't perfect, but an erroneous packet is unlikely to get through successfully. An error that just changes the bits but doesn't add or delete any has roughly a one in 64,000—which is 2^{16}—chance of going undetected. Errors that change the

number of bits are unlikely to go undetected at all. Should an error in fact go undetected, something similar might happen, depending on where in the packet the error occurred.

1. What would go wrong if we included all of these except the recipient's address in a packet? **[3 marks]**

   Solution : The network would be unable to determine the destination of the packets; so, it can't route packets to their receivers.

2. What would go wrong if we included all of these except the number of the current packet? **[4 marks]**

   Solution : The recipient will receive the packets but will be unable to put them in the right order. Most likely, it'll choose the order the packets arrive in as the "correct" order, but there's no guarantee that packets will arrive in the same order they're sent.

   Furthermore, the recipient won't know which packet is which; so, even if it knows some packet(s) didn't arrive (because it still has the total number), it won't be able to "ask for" the missing packets or say it has the received packets. So, we cannot resend dropped data (or else we always have to resend *everything* if anything is missing).

3. Finally, our protocol used acknowledgments (and "negative acknowledgments") from the recipient to the sender to ensure that packets that were lost are resent until they arrive successfully. Unfortunately, our packets were missing something necessary to make that return message possible! What is it? **[4 marks]**

   (**Hint**: the recipient needs to send a packet back to acknowledge but cannot because there's not enough information to fill in one of the four pieces of information above.)

   Solution : To send that acknowledgment back, the recipient needs to know who sent the packet, because that computer will be the recipient in the return packets.

   Unfortunately, we didn't include that!

   By the way, what if we included that *in our message* but not in the protocol information? That's the equivalent of including it inside your letter but not on the envelope in the postal system analogy. What would likely happen then? Will the system work?

   The postal system analogy should give you a pretty good guess as a starting point, by the way. The key to the answer is in the idea of "abstraction layers".

8 Critique of Poor Design [12 marks]

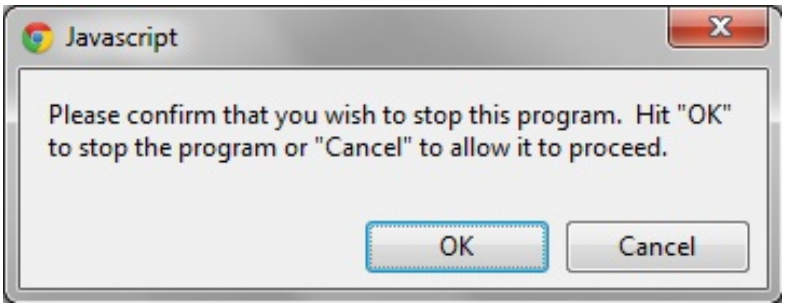
We discussed the following interface design heuristics: (a) "aim for familiarity and consistency", (b) "use well-chosen mappings and metaphors", (c) "provide useful feedback", and (d) "manage complexity". For each screenshot below, write the letter of the design heuristic that the interface shown most obviously **violates**. **Briefly justify your choices.** (**[3 marks]** per question.)

1. The result of a mistake entering the URL `www.xkcd.com/248` in a web browser.

   Violated heuristic: ____
   Briefly justify:

   Solution : (c) There's no indication of what the user did wrong or how to correct it. (Most websites now give good information, by the way. Try typing in `www.ugrad.cs.ubc.ca/blargyfoo`. You get OK information. Try the same thing on `www.cs.ubc.ca` or `www.ubc.ca`, and you get even better information. `xkcd`, by the way, is a funny and clever (but potentially offensive in some cases!) web comic written from a technical/scientific point of view.

2. A dialog box from a program.



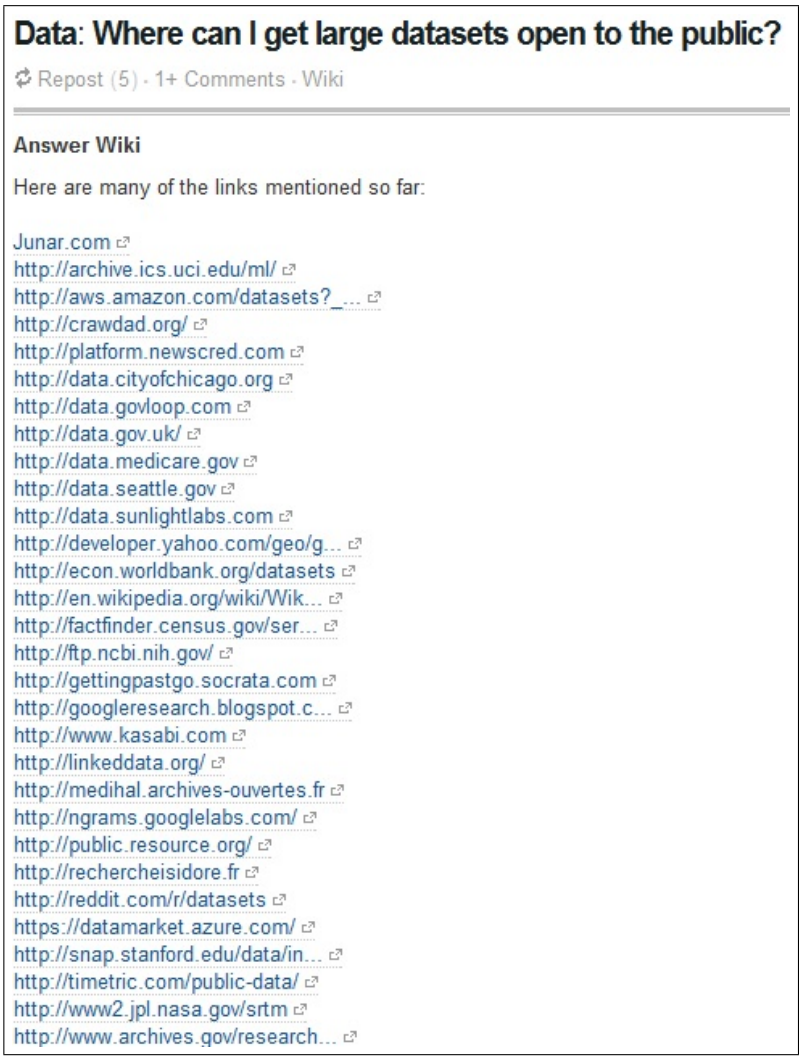
Violated heuristic: _____
Briefly justify:

Solution : (b) "Cancel" usually means "don't do this thing". It's a *bad mapping* to have it mean "go ahead and do this thing" instead. Even if it kind-of makes sense given the text of the message, users are likely to misread or not read the message because they understand what "OK" and "Cancel" means. When they make errors, is it really their fault? (If you said yes, do you read all the license agreements in software before you agree to them in order to use the software? Do you always read the "warnings" in product instructions before reading the part about how to set up or install the product, including re-reading them on products you bought before where they may have changed slightly since last time?)

3. A website listing sources for large datasets.

   Violated heuristic: _____
   Briefly justify:

   Solution : (d) There's no help for the user to scan through this large, unwieldy list. A hierarchy would be tremendously helpful! By the way, the Google ngrams data is my favourite so far. Try searching for it and play around! The main paper on this work has fascinating things to say about, e.g., Nazi suppression of artists during World War II based on quantitative data.



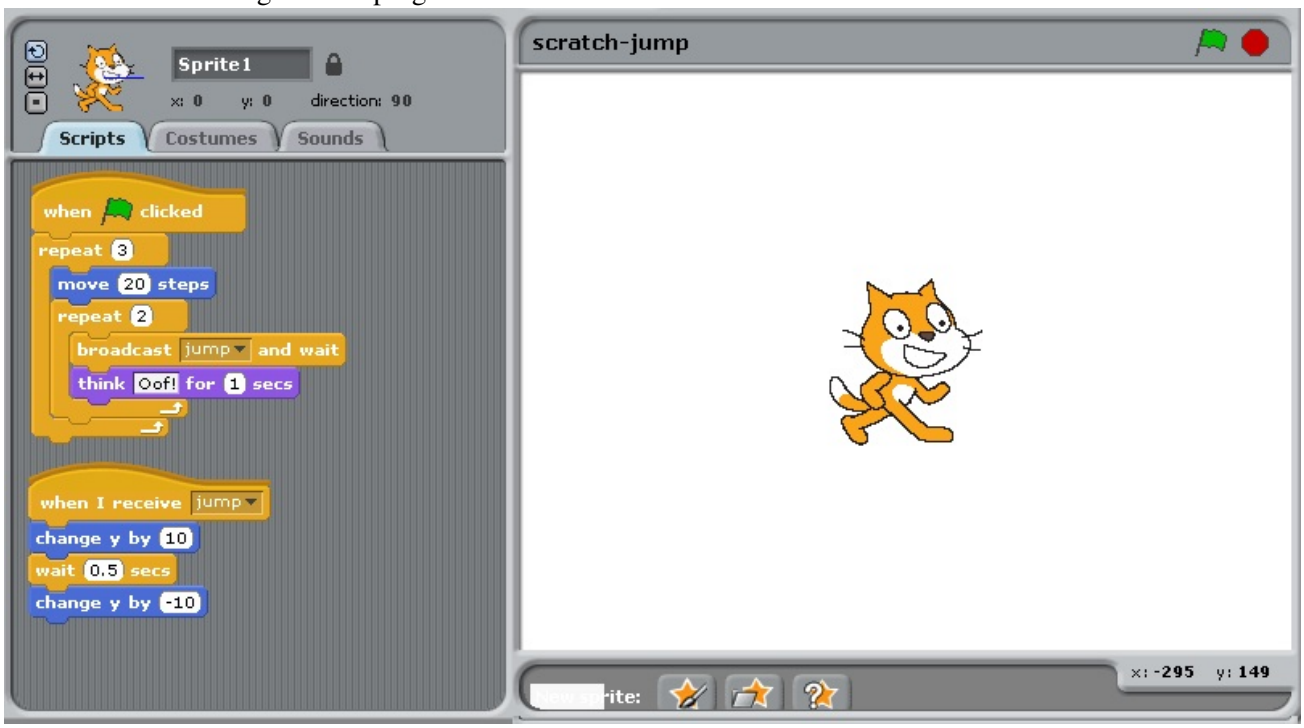
4. A cell phone keypad. (Take a close look at the arrangement of buttons.)

   Violated heuristic: _____
   Briefly justify:

   Solution : (a) Why flip the numbers top-to-bottom like this? It probably *is* a better mapping, since otherwise the placement of 0 is bizarre. But, it's not the absolute, well-established standard. So, it's likely a bad choice (unless your product is specifically differentiating itself because of this).

9 Not From Scratch [14 marks]

Consider the following Scratch program:



1. After we start the program, how many times will the cat think "Oof!" before the program finishes running? (Show your work for partial credit.) **[4 marks]**

   Solution : That command is inside a "repeat 2", which is itself inside a "repeat 3". So, the whole "repeat 2" will go 3 times. Each time, the cat will think "Oof!" twice. That's 6 times all together.

2. After we start the program, how many total steps will the cat move before the program finishes running? (Show your work for partial credit.) **[4 marks]**

   Solution : This command is only inside the "repeat 3", but it goes 20 steps. So, the cat will move 60 steps.

3. Edit the program so that the cat jumps three times as high as it does now. (If you cannot make your edits effectively above, describe them *very* clearly here.) **[6 marks]**

   Solution : In the lower block of code, change the 10 to 30 and the -10 to -30.