Problem 1)

The first interface that I could think of that I use regularly is the dashboard of the Xbox 360. It was designed as a game platform, so the typical environment it would be in is probably a living room or bedroom, possibly as a part of an entertainment center. The hardware has many flaws that make it unsuitable to that environment (it is very loud, generates a lot of heat, and is shaped awkwardly so that it can’t stack with other video equipment), but the dashboard interface at least seems to be conscious of it. It has a few nice features such as dimming the screen if it isn’t touched in a while, and it handles disconnecting and reconnecting controllers smoothly.

The 360 dashboard has several strengths. I like it that you set up profiles for each user of the system, allowing all of their saved games and control settings to be saved on a per-user basis (no more getting mad because your little brother saved over one of your precious three Zelda save slots or resetting the invert y-axis option every time you play a flight simulator). It is also nice that the dashboard uses a tabbed interface, where controls are reasonably well grouped by their functions. It is also quite nice that internet access is simple to configure and usually recognized automatically (though I’d sometimes prefer to disable it without unplugging the Ethernet cable).

Ah, but now the fun part: weaknesses. The 360 is kind of stupid when it comes to choosing which account to log in as by default. It always chooses whichever profile was set up initially, though it really ought to remember who was last logged in and use that. Similarly, in order to save a game, the system has to take a few seconds to detect all possible save devices and make you choose from a list of those available (hard drive, memory stick, etc.). I’d prefer that it default to the hard drive and let me press a button if I need to save elsewhere. Finally, the media player sucks. Until a very recent update, it couldn’t even play standard .avi files, let alone anything that requires codecs like xvid or DivX.

Another interface I use a lot is the security system from my workplace. It was designed with its environment in mind. The keypad is kept covered and somewhat camouflaged, blending into the uniform beige wall color, to deter people from messing with it. It is unobtrusive, yet mounted at eye level for most people.

The system does have a few strengths. As I mentioned, it has a hinged plate that covers the keypad when not in use, both to protect them from accidental pressing and to make the system less noticeable. It uses the standard nine-digit button layout from telephones (doesn’t sound like much, but remember that a computer keyboard reverses the ordering for no apparent reason). Each key also has brail on it, which is pretty rare for systems like this.

The problems with the system all stem from one serious design flaw- lack of feedback. If the arm code is input correctly it will beep. If it is input incorrectly, it will… also beep, just a slightly different tone. How does the first time user know if the system is armed or not? The same goes for the disarm code, where there is really no indication whatsoever if it worked or not. Would it be so hard to put a green and red LED in there labeled “armed”? Also, the keys themselves are kind of shallow and have little tactile feedback, so there is no way to know if you’ve pressed the button hard enough, at least until you’ve input the whole code and presumably heard a beep.
The last interface I’ll discuss here is the operating system for my cell phone. I’ve got the enV2 from Verizon, and they could hardly fail to design a cell phone OS wasn’t well-suited to the environment of placing and receiving phone calls.

The OS has nice features. The key guard function prevents you accidentally making calls from your pocket, yet still displays the time on the front screen whenever a key is pressed so you can still use your phone in place of a watch. The main menus options can always be navigated through via the directional pad, or you can use numeric input. You can quickly tap the volume button on the side of a phone to silence it during a call, a feature typically used when you get called in the middle of a lecture or something (if you’re silly and forgot to turn it off or set it to vibrate). Cell phones have all pretty much developed better interfaces over the last few years.

There are a few annoyances though. Certain phone settings are difficult to find, buried several levels deep under heading that seem to have nothing to do with them. The key guard settings can only be accessed using the front screen, while all other settings use the main internal screen (the phone flips open to a full qwerty keyboard, and almost never uses the front screen. Most annoyingly, your contacts list can only be accessed from the internal screen, not the external one, forcing you to open the flip-phone, choose your contact, then close it again to make speaking more comfortable. These are relatively simple software problems to fix.

Problem 2

My friend Tom had a conversation with me a little while ago about keyboards and the arbitrary things manufacturers do that ticked him off. He recently bought a new Microsoft keyboard that, by default, has the function keys mapped to do things like undo, redo, save, and the like. The problem is that they only act as the actual F1-F12 keys if you press a special “function lock” key. He was quite annoyed that he couldn’t get into his system bios until he realized where this new key was. The same keyboard also grouped the function keys into groups of three, rather than the standard four function keys per group. This was an annoying deviance from the standard that threw off his skills as a touch-typist. Finally, the wireless signal strength on many keyboards is quite poor. He was disgusted to find that he could only get a good signal if he moved the receiver on top of his desk and away from interference from his computer… almost as much trouble as having a wired keyboard in the first place. None of those issues had a big impact on his ability to get work done, but he found them frustrating and hated that a simple keyboard could force his attention away from the task at hand.

I talked to my boss for a while about different web browsers and website pet peeves. In particular, he hates Microsoft internet explorer with a passion. IE7 finally got on the tabbed browsing bandwagon, but flashes annoying “welcome to tabbed browsing” screens when you open a tab unless you dig through some settings to force it to use your home page. IE also won’t allow the little search box in the corner to open a new tab rather than displaying results in the currently active page (Firefox will do this if you set something in about:config, which is nice). The most annoying issues in his mind, though, were browser compatibility problems. Pages that work in IE often don’t work in Firefox or Safari, and vice versa. It is genuinely difficult for a web programmer to get pages to work properly in both, so they often just pick one and ignore the other. It’s very hard to quantify how much these little issues detract from the web browsing experience, but he reports that he has been much, much happier since the Google Chrome beta became available which alleviates most of his issues.
Lastly, I had a chat with my brother. His biggest problem was the upgrade from Office 2000 to Office 2007. Like a large number of disgruntled users, he hates the huge shift in user interface that Microsoft decided on. The “ribbon” interface completely changed the way anything is done in Office, to the point where finding the basic functions he needed was a big chore. Excel, in particular, proved to be nearly impossible to use because the instructions his class was given for how to use certain features was obsolete. I eventually talked him into trying OpenOffice, which he was able to adjust to easily. The ribbon also had the major disadvantage of wasting screen real estate, which was already quite limited on his laptop. The old menu-driven approach of Office 2003 and earlier may not have been perfect, but it was compact, reasonably well organized, and very familiar to its users. We were both pretty curious what we’d think of the new Office if we had never touched a previous version before. We couldn’t decide if it was the interface itself that we hated, or if it was just that the shift was so sudden and all-encompassing. Microsoft probably could have allowed for a classic mode or something, there was no need to make such a large shift permanent. It is only the GUI after all.

Problem 3)

Here’s the source:

```java
import javax.swing.JFrame;
import javax.swing.JTextField;
import java.awt.event.KeyListener;
import java.awt.event.KeyEvent;

public class PowerPractice implements KeyListener {
    public static void main(String[] args) {
        JFrame frame = new JFrame("HelloWorldSwing");
        frame.getContentPane().add(text);

        PowerPractice control = new PowerPractice();
        text.addKeyListener(control);

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
    }

    private static JTextField text = new JTextField();
    private long startTime = System.currentTimeMillis();
    private int trial = 1;
    private String isValid = "";
    private String compareText = "zyxwvuts";

    /** Most of what you need to do can be done in this method. */
    public void keyTyped(KeyEvent e) {
        String currentText = text.getText();
        //yes, this comparison is lame, but I couldn't figure out how to
        //recognize the enter key. Still works because of when text.getText
        //updates (you still have to hit enter after typing 8 letters)
        if(currentText.length() == 8){
            if(currentText.equals(compareText)){
                isValid = "valid";
            }
        }
    }
}
```
```java
} else {
    isValid = "invalid";
}
long timeElapsed = System.currentTimeMillis() - startTime;
text.selectAll();
text.replaceSelection("");
System.out.println("Trail " + trial + " was " + isValid + " with time: " + timeElapsed);
trial++;
startTime = System.currentTimeMillis();
}

/** These are not necessary, but must be included in all KeyListeners */
public void keyPressed(KeyEvent e) {}
public void keyReleased(KeyEvent e) {}
}

An here's the data I got with it:

<table>
<thead>
<tr>
<th>Trial</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16906</td>
</tr>
<tr>
<td>2</td>
<td>10953</td>
</tr>
<tr>
<td>3</td>
<td>18344</td>
</tr>
<tr>
<td>4</td>
<td>9187</td>
</tr>
<tr>
<td>5</td>
<td>7172</td>
</tr>
<tr>
<td>6</td>
<td>8250</td>
</tr>
<tr>
<td>7</td>
<td>7563</td>
</tr>
<tr>
<td>8</td>
<td>7734</td>
</tr>
<tr>
<td>9</td>
<td>7031</td>
</tr>
<tr>
<td>10</td>
<td>6640</td>
</tr>
<tr>
<td>11</td>
<td>3282</td>
</tr>
<tr>
<td>12</td>
<td>3822</td>
</tr>
<tr>
<td>13</td>
<td>5156</td>
</tr>
<tr>
<td>14</td>
<td>4172</td>
</tr>
<tr>
<td>15</td>
<td>3322</td>
</tr>
<tr>
<td>16</td>
<td>2437</td>
</tr>
<tr>
<td>17</td>
<td>2641</td>
</tr>
<tr>
<td>18</td>
<td>3297</td>
</tr>
<tr>
<td>19</td>
<td>2337</td>
</tr>
<tr>
<td>20</td>
<td>2892</td>
</tr>
</tbody>
</table>

So, using the power law of practice we get:

\[ T_n = T_1 \times n^{-\alpha} \]

\[ \frac{T_n}{T_1} = n^{-\alpha} \]

\[ -\alpha = \log_n\left(\frac{T_n}{T_1}\right) \]

Plugging in my numbers, I get \( \alpha = 0.5894 \)

The subject didn't have any real strategy, he simply tried a few times until he got a valid trial, then memorized that sequence.
Problem 4)

a)  
By Hick’s Law, RT= a+b log₂(12). A and B will need to be determined empirically.

b)  
The user would first have to pick whether they want the static or dynamic area, then pick the menu choice from one of the two areas. So you get:
\[50/50 \text{ RT} = (a+b \log_2(2)) + .5(a+b \log_2(4)) + .5(a+b \log_2(8))\]
\[75/25 \text{ RT} = (a+b \log_2(2)) + .75(a+b \log_2(4)) + .25(a+b \log_2(8))\]
\[90/10 \text{ RT} = (a+b \log_2(2)) + .90(a+b \log_2(4)) + .10(a+b \log_2(8))\]

The minimum choice time would require your choice to be in the dynamic list, because it has fewer items and should therefore be faster. That means min RT = (a+b log₂(2)) + (a+b log₂(4))

The worst of the probability splits would be the one where the static list is more likely to be used than the dynamic one, because more choices mean a slower time. Of the splits we calculated, the 50/50 one would be the worst.

c)  
Hick’s Law works best in situations that have a certain level of order to them, such as an alphabetically organized list. People mentally break down the available choices into subsections and eliminate many options at once. If a list is randomly arranged, however, scanning the list for your target is still linear, not logarithmic, so you have to be careful where you try to apply it.