#1.
Three non-desktop user interfaces: cell-phone (Samsung T809), Xbox360, and MacBook Pro

a-1) Cell-phone is used for every listed purpose. Because of the importance of mobility, the cell-phone design is focused on light-weight and small-size.

a-2) Strengths: the very small and light, the rotating camera, and the appropriate size of buttons.

a-3) Weaknesses: too short adopter cable, too small cancel key, and hard to open cover for battery.

b-1) Xbox360 is mainly used for playing games though many other functionalities are supported. So, for supporting to play games comfortably, the wireless joysticks are mainly used.

b-2) Strengths: the appropriate size and good grip of joystick, the user-friendly interface for starting game, the wireless joysticks.

b-3) Weaknesses: too big size of adopter, too big size of console, and the defective hardware.

c-1) MacBook Pro is usually used by students and professionals and they want the mobility and good spec of the laptop both. So, MacBook Pro is not so small and not so big to satisfy both needs.

c-2) Strengths: the cool design, the functions that use two fingers in trackpad, and the illuminated keyboard in low light conditions.

c-3) Weaknesses: the hot keyboard, noisy fan, no 'del' key, no eject button for dvd-rom so that only if the OS X runs well, the cd/DVD can be ejected.

#2.
1) MSN messenger for Mac

a) The video call is not supported. So, when the video call is requested, the machine should be rebooted or the virtual machine programs should be run. Sometimes, he ignores the requests for video call, when doing other works.

b) The function “Appear offline” does not work well. Sometimes, suddenly it turns to other status. If it happens when other program runs with full screen, the people who sent messages are frustrated because they think he ignores their messages. So, the function must be very excellent but it is unstable and he normally does not use it.

c) It sometimes becomes logout automatically. It is not a problem of network. It is very unstable. So, if all of my friends can moves to other messenger programs then I want to move either.
2) Desktop Computer
   a) The desktop emits heat too much. Actually, it is good for the cold season such as winter but in the summer it is very hard to endure to work with computer. It is even more serious in the small room. It is inevitable that the heat is emitted. However, more effective cooling system should be developed.

   b) The UPS (uninterruptible power supply) should be adopted in the normal supply. Of course, the crucial machines such as servers are connected and they are free from the interruption of electric power. However, the computers for home or small business are not. Because of the unexpected interruption, very crucial data can be gone without save. The power supply should have the battery to sustain to give time for saving important files at least.

   c) It is frozen very often. So, Some applications have auto-save function. However, we do not usually set up it as very short period, because some of them disturb our work and concentration.

3) Xcode and Interface Builder
   a) The Xcode supports much lower level of auto-completion function than Microsoft Visual Studio. The function helps developers know what kinds of method exist for the specific class very easily. However, because of the lack of auto-completion for Xcode, the developers spend a lot of time to search the methods from the Internet.

   b) The beginner who uses Interface Builder can hardly know the instance name such as buttons that they made. So, the interface of Interface Builder should be more developer-friendly.

   c) When the project is complied and error or warning exists, the complier shows us same message twice. It does not cause a big problem, but it is useless. It seems there are more errors or warnings than really there exist.

#3.

<table>
<thead>
<tr>
<th>trials</th>
<th>input by user</th>
<th>correctness</th>
<th>spent time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>21.611525</td>
</tr>
<tr>
<td>2</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>7.851141</td>
</tr>
<tr>
<td>3</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>14.465954</td>
</tr>
<tr>
<td>4</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>5.37405</td>
</tr>
<tr>
<td>5</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>5.658846</td>
</tr>
<tr>
<td>6</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>17.645887</td>
</tr>
<tr>
<td></td>
<td>zyxwvuts</td>
<td>Correct</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>7</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>4.958218</td>
</tr>
<tr>
<td>8</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>7.59728</td>
</tr>
<tr>
<td>9</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>6.363348</td>
</tr>
<tr>
<td>10</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>4.687629</td>
</tr>
<tr>
<td>11</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>5.240261</td>
</tr>
<tr>
<td>12</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>9.944364</td>
</tr>
<tr>
<td>13</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>4.926493</td>
</tr>
<tr>
<td>14</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>6.9941</td>
</tr>
<tr>
<td>15</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>5.143997</td>
</tr>
<tr>
<td>16</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>4.733999</td>
</tr>
<tr>
<td>17</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>5.876127</td>
</tr>
<tr>
<td>18</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>5.458185</td>
</tr>
<tr>
<td>19</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>4.267262</td>
</tr>
<tr>
<td>20</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>4.305146</td>
</tr>
<tr>
<td>21</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>3.653265</td>
</tr>
<tr>
<td>22</td>
<td>zyxwvuts</td>
<td>Correct</td>
<td>4.866205</td>
</tr>
</tbody>
</table>

\[ T_n = T_1 \times n^{(-\alpha)} \]
4.866205 = 21.611525 * 22^{(-\alpha)}
\alpha = 0.482333

(Interface of the program)

(Source Code: Cocoa Objective-C)

<AppController.h>
#import <Cocoa/Cocoa.h>
@interface AppController : NSObject {
    IBOutlet NSTextField *inputTextField;
    IBOutlet NSTextField *outputTextField;
    int leftTimes;
    int trialsNumber;
    NSTimeInterval startTime;
    NSTimeInterval endTime;
    NSMutableString *mutstr;
}
-(IBAction)submit:(id)sender;
@end

<AppController.m>
#import "AppController.h"

@implementation AppController

- (void)awakeFromNib
{
    leftTimes = 22;

    NSString *message = [[NSString alloc] initWithFormat:@"Please correctly input in %d times more.\n", leftTimes];
[outputTextField setStringValue:message];
[message release];

startTime = [NSDate timeIntervalSinceReferenceDate];
mutstr = [[NSMutableString alloc] init];

-

(IBAction)submit:(id)sender {
NSString *output;
NSString *inputText = [inputTextField stringValue];
NSString *correctAnswer = @"zyxwvuts";
NSTimeInterval timeInterval;
NSError *error;

// increase the number of trials
trialsNumber++;

// get the input interval
derTime = [NSDate timeIntervalSinceReferenceDate];
timeInterval = endTime - startTime;

if (leftTimes == 0)
{
    // write to file
    [mutstr writeToFile:@"Data.txt" atomically:YES encoding:NSUnicodeStringEncoding error:&error];

    // terminate application
    [NSApp terminate:sender];
}
else if (leftTimes > 0)
{
    // the testee correctly input
    if ([[inputText isEqualToString:correctAnswer]])
    {
        leftTimes = leftTimes - 1;
        if (leftTimes == 0)
        {
            output = [[NSString alloc] initWithFormat:@"Done!! Thanks you for your cooperation"];
            // disable the input text field
            [inputTextField.setEditable:0];
            [inputTextField setSelectable:0];
        }
    }
    else

Actually, my friends does not input at the very end. He typed “stuvwxyz” order but also pushed <- arrow after typing each character. Eventually, the result is same as the correct answer and he can save pretty much time.

#4. Assume that $A = 548, B = 420$

(a) $MT = A + B \cdot \log_2(n)$

$$= 548 + 420 \cdot \log_2(12)$$

$$= 2053.68\text{ms}$$
(b) 50/50 ratio:
\[ 0.50 \times (548 + 420 \times \log_2(4)) + 0.50 \times (548 + 420 \times \log_2(8)) = 1598\text{ms} \]
75/25 ratio:
\[ 0.75 \times (548 + 420 \times \log_2(4)) + 0.25 \times (548 + 420 \times \log_2(8)) = 1493\text{ms} \]
90/10 ratio:
\[ 0.90 \times (548 + 420 \times \log_2(4)) + 0.10 \times (548 + 420 \times \log_2(8)) = 1430\text{ms} \]

As the ratio for the static decrease, the choice time decrease.
So, the minimum choice time occurs: **100% for the dynamic menu**
\[ 548 + 420 \times \log_2(4) = 1388\text{ms} \]

Also, the maximum choice time occurs: **100% for the static menu**
\[ 548 + 420 \times \log_2(8) = 1808\text{ms} \]

(c) The Hick's Law requires two empirical constant A and B. That means that if A and B cannot be calculated by experience or observation, Hick's Law cannot be used and there exist lots of cases that simply cannot get the numbers. So, it is limited to apply in the real-world performance.