# 軟體工程實務 Lab2 - 簡易計算機 Bug 說明文件

學號：113522118
姓名：韓志鴻

1. **Bug：**在按完算式並連續點擊等號按鍵後，要能夠連續處理最後一個運算操作。例如預設測資中的 120＋9，當連續按下等於後，每次都要再執行+9 的操作。此預設測資中連續按了 5 次等號，故應 120＋9＋9＋9＋9＋9＝165 才正確，但實際操作後答案變成 609。

   **Solution：**追蹤程式碼中的 equals_Click 函式(圖 1)，在原寫法中當點擊第 2 次等號之後，會把當前顯示的數字當成 num2，但 num1 沒變。因此以預設測資而言，第 1 次等號後變成 120＋9＝129，但後 4 次每次都會加 num1 = 120，因此答案才會變成一開始的 609。所以要寫一個 if-else(圖 1 紅框部分)，**當 currentNum == 1 成立，代表當前運算僅按下等號按鍵，此時計算機畫面上的加總值作為 num1 而 num2 保持不變；當 else 成立，代表當前運算有實際按了加減乘除導致改變了 currentNum 的值(加減乘除會讓 currentNum 變成 2)，此時就是正常的運算**。另外也要修改 currentNum 的範圍(圖 1 紅底線)，將其控制在 1 和 2 來區分實際按運算符號還是連續按等號。

2. **Bug：**若在 no input 下直接按運算符號會直接彈出報錯訊息(圖 2)，理想情況下此時計算機應該不要有所反應。

   **Solution：**新增 textBox1.Text.Length > 0 && textBox1.Text != "."做為防呆機制(圖 1 黃底線)，**讓運算符號不能在計算機顯示區無任何數字或只有小數點之下有所反應**。同時此條件也新增在加減乘除的函式內(圖 3)。

3. **Bug：**計算機不能在按下等號後繼續作其他運算。

   **Solution：**在等號函式內**引用原有的 button_show 函式**(圖 1 藍底線)，該函式可解鎖加減乘除按鈕，讓計算機在按下等號後還可進行其他運算操作。

4. **Bug：**小數點可重複輸入。

   **Solution：**如圖 4，**新增!textBox1.Text.Contains(".")條件**，若計算機畫面上沒有小數點才可輸入。

```
private void equals_Click(object sender, EventArgs e)
{
    if (textBox1.Text.Length > 0 && textBox1.Text != ".") // 新增
    {
        if (currentNum == 1) // 新增
        {
            num1 = Convert.ToDouble(textBox1.Text); // 新增
        }
        else // 新增
        {
            num2 = Convert.ToDouble(textBox1.Text); // 原寫法
        }

        switch (op)
        {
            case "+":
                result = num1 + num2;
                break;
            case "-":
                result = num1 - num2;
                break;
            case "*":
                result = num1 * num2;
                break;
            case "/":
                result = num1 / num2;
                break;
        }
        textBox1.Text = Convert.ToString(result);
        currentNum = 1; // 原：currentNum++
        button_show(); // 新增
        return;
    }
}
```

▲ 圖 1：等號函式內部程式碼修改


▲ 圖 2：無輸入直接按加號會出現錯誤訊息，其他運算符號同理

```
private void addition_Click(object sender, EventArgs e)
{
    if (currentNum == 1 && textBox1.Text.Length > 0 && textBox1.Text != ".") // 新增：&& textBox1.Text.Length > 0 && textBox1.Text != "."
    {
```

```
private void subtraction_Click(object sender, EventArgs e)
{
    if (currentNum == 1 && textBox1.Text.Length > 0 && textBox1.Text != ".") // 新增：&& textBox1.Text.Length > 0 && textBox1.Text != "."
    {
```

```
private void multiplication_Click(object sender, EventArgs e)
{
    if (currentNum == 1 && textBox1.Text.Length > 0 && textBox1.Text != ".") // 新增：&& textBox1.Text.Length > 0 && textBox1.Text != "."
    {
```

```
private void division_Click(object sender, EventArgs e)
{
    if (currentNum == 1 && textBox1.Text.Length > 0 && textBox1.Text != ".") // 新增：&& textBox1.Text.Length > 0 && textBox1.Text != "."
    {
```

▲ 圖 3：如同圖 1 黃底線，在加減乘除函式內做相同的防呆機制

```
private void decimalPoint_Click(object sender, EventArgs e)
{
    if (!textBox1.Text.Contains(".")) // 新增：若textBox1.Text中不含 "."
    {
        textBox1.Text += ".";
    }
}
```

▲ 圖 4：小數點函式內部程式碼修改