Final Project

根據題目說明，code-11 可以區分為輸入、解碼與校驗三個部分。經過討論，我們決定把專案分成以下步驟 ：

1. 讀取資料與檢查資料的有效與否
2. 將所有的 d 轉為二進位值
3. 將 barcode 轉正
4. 將二進位值轉為字串
5. 檢查校驗碼 C 與 K
6. 顯示結果

一、讀取資料

依據題目說明$m \leq 150$ 以及$1 \leq d_i \leq 200$, 所以我們定義了`int dataSize` 以及一個`FILE* inputFile`讀取資料。

```
printf("Enter file name :");
scanf("%s", fileName);
inputFile = fopen(fileName, "r");
if (!inputFile) {
    printf("Can\'t find it.");
} else {
    while ((fscanf(inputFile, "%d", &dataSize)) && dataSize != 0) {
        if (dataSize < 23 || (dataSize + 1) % 6 != 0) {
            printf("Case %d: ", caseNumber);
            printf(BAD_CODE);
        } else {
            charSize = (dataSize + 1) / 6;
            for (int i = 0; i < dataSize; i++) {
                fscanf(inputFile, "%d", data + i);
            }
        }
    ...
    }
}
```

其中，在第四行的`if (dataSize < 23 || (dataSize + 1) % 6 != 0)`中我們判斷了m是否為有效地輸入。因為根據題目說明，輸入的測資並不包含長度為 0 的訊息。因此，輸入的m應該要至少大於 23，也就是開始、C、K、結束，以及各訊息間的間隔的總和。

$$m \geq 5 \times (start + C + K + end) + 4 = 24$$

除此之外，我們可以將所有資料看作 6bits 為一個單位(除了最後的 end 之外)，因為每個訊息之後都帶有一個間隔，於是我們設想如果輸入是正確的，那麼$m + 1 \equiv 0(mod(6))$應該會成立。於是，if 的成立意味著輸入的測資應是無效的，應該直接跳至步驟六並顯示出 bad code.

二、將所有的 d 轉為二進位值

在這個步驟中，我們首先需要將所有的d判斷出它的二進制應為 0 或 1。在此，我們找出所有d的最大值及最小值並求平均，得到了 0 與 1 的判斷標準。`boundary = (float)(max + min) / 2;`

在將它們轉換之前，我們還需要對d進行驗證，確保它們在$\pm 5\%$的範圍內，否則應該要跳至步驟六並顯示出 bad code. 為求效率，我們知道了寬條的寬度是窄條的兩倍。所以，我們將所有的窄條乘以二便能一同與寬條進行判斷，節省時間。接著，我們再次對d求出最大及最小值，並根據以下數學原理進行判斷，檢查是否有任何d超出誤差範圍。

$d_{max} \times 0.95 = N_1, \quad d_{min} \times 1.05 = N_2, \quad X$*為標準寬度*

$\because X \le d_{max} \le 1.05X, \quad 0.95X \le d_{min} \le X$

分別乘以 0.95、1.05, *則* $0.95X \le N_1 \le 0.9975X, \quad 0.9975X \le N_2 \le 1.05X$

$\therefore N_1 \le N_2$

如果$N_1 \le N_2$不成立，則表示存在某個d超出了誤差範圍。確認完d在誤差內之後，我們則依照先前的標準，將d轉換為二進制回傳。

三、將 barcode 轉正

由於 Start/Stop 的編碼都為 00110，可以由第二個位元得知整個條碼是否顛倒。因此，我們以第二個位元為判斷標準，如果為 1 就將條碼反轉。

```
if (*(binary + 1) == '1') {
    reverse(binary);
}
```

如此一來，我們就得到了正向的條碼，接下來就可以開始對它進行解碼。

四、將二進位值轉為字串

我們利用了步驟一所說過的除了最後的 Stop，其他資料可以視為六個位元一組。將 Stop 的後面也加上一個 0，這樣我們就能直接將資料統一為六個位元一組，並將他們解碼。解碼方式則是直接比較字串，依據字串回傳相應的答案。無法解碼時，則傳回 n。並在之後的處理中視為解碼失敗，跳至步驟六並顯示 bad code.

```
char bits6_to_char(char* bin) {
    if (bin[5] == '0') {
        bin[5] = '\0';
        if (!strcmp(bin, "00011")) return '0';
        else if (!strcmp(bin, "01001")) return '1';
        else if (!strcmp(bin, "10000")) return '2';
        else if (!strcmp(bin, "10100")) return '3';
        else if (!strcmp(bin, "11000")) return '4';
        else if (!strcmp(bin, "00101")) return '5';
        else if (!strcmp(bin, "01100")) return '6';
        else if (!strcmp(bin, "00001")) return '7';
        else if (!strcmp(bin, "10010")) return '8';
        else if (!strcmp(bin, "10001")) return '9';
        else if (!strcmp(bin, "00100")) return '-';
        else if (!strcmp(bin, "00110")) return 'B';
        else
            return 'n';
    } else {
        return 'n';
    }
}
```

五、檢查校驗碼 C 與 K

最後，我們依照題目給的公式定義了一個函數

`int checkCandK(char* decodeData, int size, char* anser)`來檢查 C 與 K 是否正確並回傳解碼資料、bad C. 或 bad K. 給`anser`。

六、顯示結果

依照步驟五所給出的答案、或在之前就因為錯誤而跳至此步驟的 bad code. 顯示於螢幕上。

```
printf("Case %d: ", caseNumber);
printf("%s", anser);
```

七、附錄(source code)

https://github.com/fili-asphalt/ComputerPorgramming_Final

八、分工

學號	姓名	工作分配
B10915017	楊笙宏	步驟一、二、四 word 技術支援
B10915021	王靖	步驟三、五 報告撰寫

九、心得

1. 王靖

在這次的專題，我們分析完題目之後採取自己領工作的模式，對已經有想法的部分先行下手，並將它們寫成 function 讓 main 中的程式盡可能的直觀，增加可讀性。整個專題的進行過程對我們造成最大的阻礙是判斷 5%誤差的部分。在討論的過程中，我們原本有兩個可行的方法: 第一個是由現有的資料推算出誤差範圍，再尋找是否有超出範圍的資料。第二個是藉由判斷最大最小值之間的關係確認是否有資料超出範圍。雖然兩種方法都確定可行，但後者的實現較為容易許多，因此最後選擇了後者。

我在製作專題的時候，感受到了分工合作所帶來的效率。每個人都專注於自己的小部分，不僅降低了作業難度，也提高了程式的整合性。我很樂於使用這樣的開發模式，也了解了這種模式下所帶來的優點。

2. 楊笙宏

這次專題改變了我的思考方式，以往遇到一個問題我總會依照直覺去想，想到什麼寫什麼，只是單純的暴力破解，但是這道題目無法這樣做，因為當中牽涉了概念較為複雜的邏輯，我認為這道題目最關鍵的點是判斷寬度是否在合理範圍內的演算法，是設計過程中遇到最大的問題，需要通過數學的計算及證明才能寫出這項演算法，若是單純依靠直覺思考，那邏輯會混雜在一起，邏輯發生錯誤便很難從中找出錯誤點；經過數學或其他工具輔助的思考方式則可以清楚地檢視每一道邏輯，兩者相比後者顯得更有效率，因此這種思考方式在解決問題時幫助了我很多，在解決問題的能力方面也增長了不少。