模擬練習 2

開始時間:_____、結束時間:_____

完整解題:

Q1: _____ Q4: _____

Q2: _____ Q5: _____

Q3: _____ Q6: _____

試題一：局部編碼(17 分)

說明：局部編碼有許多應用，局部編碼(*LC*)是在一個資料區域內，利用相鄰鄰居和不同權重，來進行編碼。局部編碼可以利用下述公式來表示：

$$LC_{I,R}(x_c,y_c)=\sum_{i=0}^{I-1}T(d_i-d_c)\times 2^i \qquad (1)$$

$$T(x)=\begin{cases}1 & x\geq 0\\ 0 & x<0\end{cases} \qquad (2)$$

其中，d_c是在區域(R)中的中心點(x_c, y_c)之資料，d_i是區域中心點之 I 個鄰居資料點，d_i-d_c表示鄰居資料點與中心點之差，2^i是鄰居資料點之對應權重，$T(x)$是一閥值函數，當 x 大於等於 0 時，$T(x)$=1，當 x 小於 0 時，$T(x)$=0。

程式功能：請利用上述公式(1)和公式(2)，寫一個程式，能完成以下功能之要求：

(1)能讓使用者輸入 6x6 資料，這些資料要大於等於 0，不可以小於 0。

(2)能讓使用者輸入 3x3 權重，這些權重是 2 的次方。

(3)能讓使用者輸入要編碼之 3x3 區域的左上角座標。

(4)程式可以計算和顯示要編碼之 3x3 區域的編碼結果。

程式執行範例：

範例說明:從上圖左邊開始，第 1 步讓使用者輸入 6x6 資料，第 2 步讓使用者輸入 3x3 權重，第 3 步讓使用者輸入要編碼之 3x3 區域的左上角座標，座標請參考輸入資料之座標(0~5, 0~5)，第 4 步按計算執行，第 5 步顯示編碼結果。

上述範例，權重遮罩設定為 $\begin{bmatrix} 1 & 2 & 4 \\ 8 & 0 & 16 \\ 32 & 64 & 128 \end{bmatrix}$，輸入要編碼之 3x3 區域的左上角座標為(1, 1)，也就是要對輸入區域資料 $\begin{bmatrix} 55 & 35 & 28 \\ 52 & 43 & 38 \\ 26 & 65 & 46 \end{bmatrix}$ 進行編碼。再利用公式(2)運算後，其結果為 $\begin{bmatrix} 1 & 0 & 0 \\ 1 & & 0 \\ 0 & 1 & 1 \end{bmatrix}$。再利用公式(1)運算後，其編碼結果為 201，如上圖所示。

若妳（你）的程式都完成上述功能和要求，才可以上傳要求檢查功能。

試題二：設計一分區停電程式(17分)

說明:為了解決未來電力短缺問題，電力公司將採取分區停電方案。此方案的作法為先將台灣分為N個區，接著挑選一個數字，m。停電先由區域1 開始，然後每隔m個區為下一個停電區，超過最後一區域N，則再由區域1重新接下去。不過為了公平起見，計算m個區域時只會將尚未停過電的區域算進去，已經停過電的區域不算在內。以N = 17 而 M = 5為例，停電順序依序為1, 6, 11, 16，由於16 + 5超過17，所以從頭由區域1接續下去，同時由於區域1已經停過電，不予計算，因此下一個區域計算方法為17, 2, 3, 4, 5，所以區域5為下個停電區域。再來由於區域6和11都停電過，所以隔五個沒停過電的區域7, 8, 9, 10, 12，得知區域12接在區域5之後。以此類推，可以算出分區停電順序為1, 6, 11, 16, 5, 12, 2, 9, 17, 10, 4, 15, 14, 3, 8, 13, 7。如M=6, 停電順序依序為 1, 7, 13, 3, 10,17, 9, 2, 12, 6, 4, 16, 5, 11, 8, 15, 14。

針對特定的分區數目N，不同的數字M會導致最後停電區域不同，由於台電總公司位於台北(13區)，**13區必須為最後停電區域，因此M值必須謹慎選擇**。以N = 17為例，如果選擇M=7，停電順序為1, 8, 15, 6, 14, 7, 17, 11, 5, 3, 2, 4, 10, 16, 9, 12, 13，才能使得13區為最後停電區域。這樣的M數字或許不止一個，但是**我們只需求出13區為最後停電區域中最小的一個M值**。

請設計一程式讀取電力分區數N(最小為13，最大為99，13≦N≦99)後，然後M由1, 2, 3, …漸增直找到某一M數字使得13區為最後停電之區域，該M數字即為答案，請將該M數字輸出。

例如輸入N=17時，輸出為 M=7。

試題三：在地圖中搜尋最低成本的路徑。(17分)

說明：1. 如圖 3-1 所示為一交通示意圖，圓圈節點代表城市，並以數字表示其順序；連接線表示城市間的行進方向，連接線的數字代表成本(成本=距離*時速)。某君規劃假期將由出發地到目的地(如示意圖，由節點①到節點⑦)，假設城市間的各路段之距離及時速可事先查詢完成，則某君在出發前即可將各路段之距離及時速換算為成本(如示意圖)，據以搜尋最低成本的路徑，並處理得知其所經各城市節點的先後次序，及其路徑的總和最低成本值。請設計一程式完成之。**範例一：**

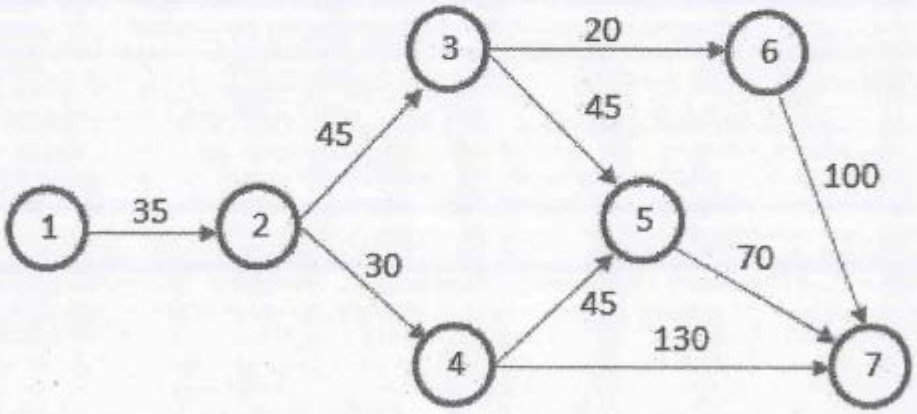


圖 3-1

2. 當城市間連接線的成本值改變，仍然可搜尋最低成本的路徑，並獲得其所經各城市節點的先後次序，及其路徑的總和最低成本值。

輸入格式：1. 依序以「**城市編號　城市編號　城市間連接線的成本數字**」表示，例如：由節點①到節點②的表示為「1　2　35」。餘此類推，將所有城市及城市間連接線的數字設定完成，並存成輸入檔。

2. 輸入檔內容可隨時更改，格式如圖 3-2 所示

```
1 2 35
2 3 45
2 4 30
3 5 45
3 6 20
4 5 45
4 7 130
5 7 70
6 7 100
```

圖 3-2

```
最低成本值總和:180
路徑次序:  1  2  4  5  7
連線數值:  0 35 30 45 70
```

圖 3-3

3. 可隨時更改輸入檔的成本值，再搜尋最低成本的路徑次序及其最低成本值。

輸出格式：印出最低成本值總和、其所經各城市節點的先後次序及其路徑值，如圖 3-3 所示：

範例二：1. 節點⑥到節點⑦的值改為 70，如圖 3-4 所示：

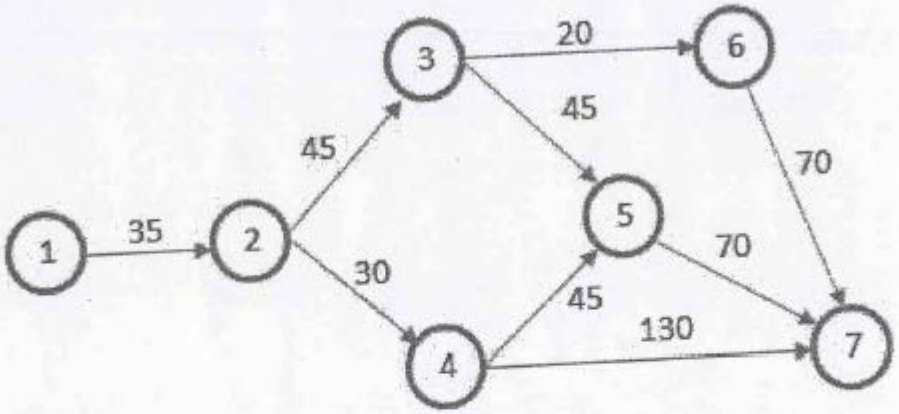


圖3-4

```
1 2 35
2 3 45
2 4 30
3 5 45
3 6 20
4 5 45
4 7 130
5 7 70
6 7 70
```

輸入格式：輸入檔更改內容，如上所示

輸出格式：印出最低成本值總和、其所經各城市節點的先後次序及其路徑值，如下所示：

```
最低成本值總和:170
路徑次序:  1  2  3  6  7
連線數值:  0 35 45 20 70
```

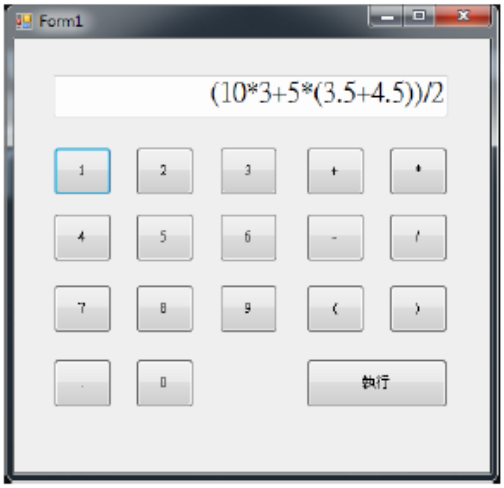
題目四：四則運算機

說明：

設計一簡易型計算機介面，使用者從該介面輸入四則(加減乘除)運算式後，程式需將原四則運算式與正確結果輸出至檔案(b.txt)中。輸入之四則運算式中需包含有括號以改變運算之優先順序，且可輸入不只一層括號，輸入之數值需可包含浮點數。如輸入之四則運算式含有其他無法辨識的符號(例如 a、c、$、%、...等)，或運算式不合規範(括號數錯誤、括號順序錯誤、運算子錯誤、浮點數格式錯誤、...等)，則程式需顯示”運算式有誤!”字樣，且不會產生 b.txt 檔案。

參考範例一：

輸入四則運算式：(10*3+5*(3.5+4.5))/2



執行後，程式會自動產生一 b.txt 檔案，並於 b.txt 檔案中輸出：(10*3+5*(3+5))/2 = 35

註：b.txt 檔案輸出之方程式中，”=”前後需各含有一個空格。

參考範例二：

輸入四則運算式：(2.5*4-2+5%(2.7+5.3))/2

程式執行後，系統顯示”運算式有誤!”字樣，且不會產生 b.txt 檔案。

提示：

平常所使用的運算式，主要是將運算元放在運算子的兩旁，例如：a+b/d，這樣的式子，這稱之為中序（Infix）表示式。對於人類來說，這樣的式子很容易理解，但由於電腦執行指令時是有順序的，遇到中序表示式時，無法直接進行運算，而必須進一步判斷運算的先後順序，所以必須將中序表示式轉換為後序（Postfix）表示式，後序表示式又稱之為逆向波蘭表示式（Reverse polish notation）。例如：(a+b)*(c+d)這個式子，後序表示式為：ab+cd+*，轉換演算法的輸出過程如下：

OP	STACK	OUTPUT
(	(	-
a	(	a
+	(+	a
b	(+	ab
)	-	ab+
*	*	ab+
(	*(	ab+
c	*(	ab+c
+	*(+	ab+c
d	*(+	ab+cd
)	*	ab+cd+
-	-	ab+cd+*

試題五：實數與二進位數之編碼解碼（16 分）

請撰寫一程式，根據實數變數 x 的範圍(range)以及精確度(precision)的要求，程式必須能夠將使用者所輸入之實數轉換成相對應之二進位數，或是將二進位數轉換成相對應之實數。

說明：

我們希望使用二進位向量(binary vector)來代表一實數變數 x，該二進位向量的長度很顯然與所需要的精確度相關。

假設變數 x 的範圍是[-1.0　2.0]，編碼的精確度為小數點以下 6 位，則[-1.0　2.0]需要切割成：

$(2.0-(-1.0))\times 10^6 = 3000000$ 個等分，這表示我們需要使用 22 位元來進行編碼：

$$2097152 = 2^{21} < 3000000 < 2^{22} < 4194304$$

以得到二進位字串：

$$(b_{21}b_{20}b_{19}\ldots\ldots b_0)$$

亦即：

二進位字串$(000\ldots\ldots 0)$將代表實數值-1.0，$(111\ldots\ldots 1)$ 將代表實數值 2.0。

如果要進行解碼，由一個二進位字串轉換成一個實數值 x，作法如下：

1. 將二進位字串：

$$(b_{21}b_{20}b_{19}\ldots\ldots b_0)$$

由基底 2 轉換為基底 10：

$$(b_{21}b_{20}b_{19}\ldots\ldots b_0)_2 = (\sum_{i=0}^{i=21} b_i\cdot 2^i)_{10} = x'$$

2. 求得對應之實數值 x：

$$x = -1.0 + x'\cdot\frac{(2.0-(-1.0))}{2^{22}-1}$$

範例一：假設實數變數 x 的範圍是[-1.0　2.0]，編碼的精確度為小數點以下 6 位。

則二進位字串

$$(1000101110110101000111)$$

即是代表實數值 0.637179，因為

$$x' = (1000101110110101000111)_2 = 2288967_{10}$$

$$x = -1.0 + 2288967\cdot\frac{(2.0-(-1.0))}{2^{22}-1} = -1.0 + 2288967\cdot\frac{3}{4194303} = 0.637197$$

範例二：假設變數 x 的範圍是[-3.0　12.1]，編碼的精確度為小數點以下 4 位，則我們需要以 18 位元來進行編碼。

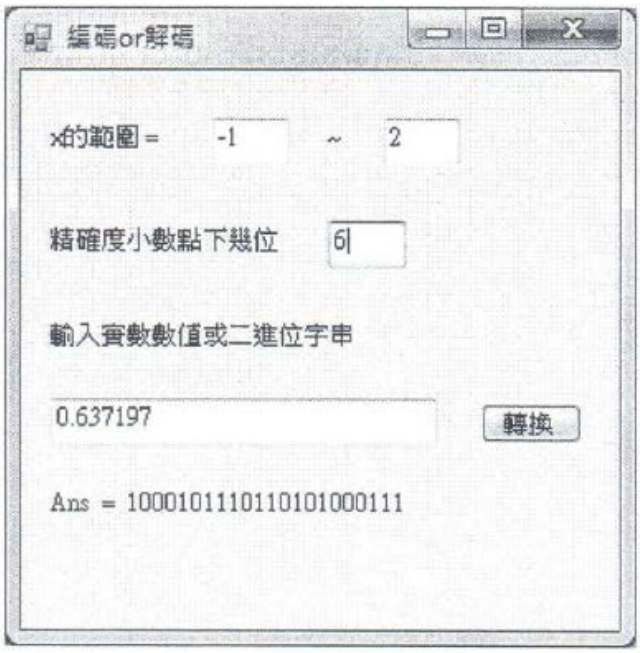
因此，二進位字串 ：(010001001011010000)

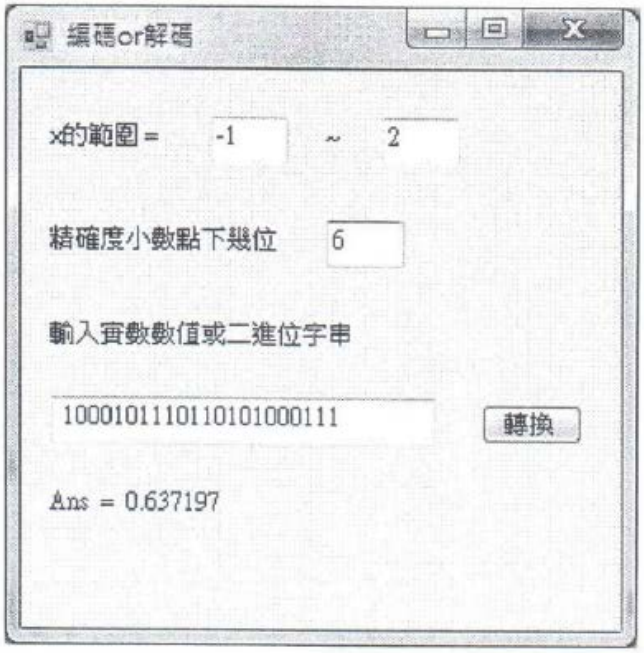
即是代表實數值 1.0524，因為

$$x = -3.0 + (010001001011010000)_2\cdot\frac{(12.1-(-3.0))}{2^{18}-1} = -3.0 + 70352\cdot\frac{15.1}{262143}$$

$$= 1.0524$$

程式執行範例和要求:當程式剛執行時會出現一視窗,並提示使用者先輸入實數變數x的範圍，以及編碼的精確度(小數點以下位數)，程式首先需要求得編碼所需之位元數。當使用者輸入一實數值之後，程式需能自動轉換該實數值成相對應之二進位字串；當使用者輸入一個二進位字串之後，程式需能自動轉換該二進位字串成相對應之實數，如下圖所示。





試題六：迷宮遊戲: (16 分)

說明：以二維陣列模擬的迷宮地圖存放在檔案(.txt)中，如圖 6-1 範例所示：
其大小為 8*8，其中"1"代表不通的路，"0"代表可通的路，此迷宮包含至少一條能夠從入口到出口的路徑。將一老鼠放入迷宮入口處(0,0)，令其尋找出口(7,7)，程式須讀取使用者指定之不同 8*8 迷宮地圖檔(xxx.txt)，計算老鼠所有經過的路徑，並輸出在螢幕上。
老鼠的移動過程需依以下的規則進行：

1. 老鼠一次只能走一格，可以有八個不同的方向嘗試移動下一步，分別是北(N)、東北(NE)、東(E)、東南(SE)、南(S)、西南(SW)、西(W)、西北(NW)。
2. 老鼠移動的方向，必須依照以下順序：北(N)、東北(NE)、東(E)、東南(SE)、南(S)、西南(SW)、西(W)、西北(NW)。
3. 遇到無路可走時，需退回一步尋找其他可行的路徑。
4. 已走過的路不能再走第二次。

提示:當路徑走到盡頭卻不是迷宮出口時，則回到上一個叉路口，再選擇沒走過的路前進。亦即，藉由不斷地的回溯(backtracking)，也就是在錯誤發生時回到上一個狀態，以嘗試另外一個選擇，直到找到出口。

範例：圖 6-1 是一個 8*8 迷宮，程式執行後，可以獲得老鼠所有經過的路徑為如圖 6-2:

```
入口→  01110110
       00010001
       01011011
       11000010
       10011010
       11001011
       11111010
       00111100 →出口
```

圖 6-1

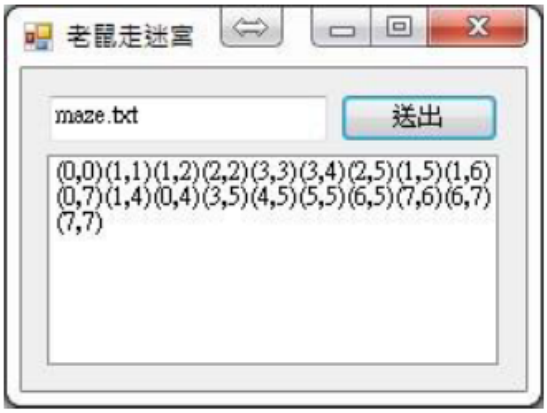


圖 6-2

第一次路徑(錯誤解)

```
入口→ 01110110
       00010001
       01011011
       11000010
       10011010
       11001011
       11111010
       00111100 →出口
```

經過第一次回溯後再嘗試之路徑(錯誤解)

```
入口→01110110
       00010001
       01011011
       11000010
       10011010
       11001011
       11111010
       00111100 →出口
```

經過第二次回溯後再嘗試之路徑(正確解)

```
入口→01110110
       00010001
       01011011
       11000010
       10011010
       11001011
       11111010
       00111100 →出口
```