

### 上機習題 #3 (stack 之應用)

題目：西洋棋騎士(knight)走完棋盤的順序

說明：

一個  $n \times n$  的西洋棋盤上，騎士(knight)置於某位置(x,y)，撰寫一個程式以便找出此騎士( $n^2-1$ )次移動的順序，而每個格子只拜訪過一次。實作方式使用 stack，而不使用 recursion。西洋棋騎士可能移動之方位共有 8 個如下：

		→				
		1	2	3	4	5
↓	1		K8		K1	
	2	K7				K2
	3			K		
	4	K6				K3
	5		K5		K4	

上圖中，K 代表目前騎士所在之位置，K1 至 K8 為可能移動之位置。假設 K 所在之位置為(i,j)，則移動後之位置差值如下：

	i 之差值	j 之差值	位置
K1	-2	1	(i-2,j+1)
K2	-1	2	(i-1,j+2)
K3	1	2	(i+1,j+2)
K4	2	1	(i+2,j+1)
K5	2	-1	(i+2,j-1)
K6	1	-2	(i+1,j-2)
K7	-1	-2	(i-1,j-2)
K8	-2	-1	(i-2,j-1)

可以利用嘗試錯誤的方法(backtracking)解決此問題。騎士在某位置時，可以選擇某條路往前走，走到不能繼續前進時（所有可能移動的位置，均已嘗試過），則退回上一個位置，再選擇其他路徑。由於

騎士往前走，而將來又可能退回，故可利用 `stack` 記錄騎士所走過的路徑。

Stack 的每一個元素應至少記錄三項資料如下：

i 軸之值、j 軸之值、該位置已經走過的方向編號

例如：n=3

initial:

0	0	0
0	0	0
0	0	0

從左上角(1,1)出發，3 步之後：

1	0	0
0	0	2
3	0	0

上圖的 2 與 3 分別表示第二次與第三次拜訪的位置，0 表示尚未拜訪過。目前處於第三步，此時的 stack 內容如下：

2,3,6
1,1,3

Stack 的內容：每一筆紀錄代表棋盤被走過的狀態。從第二步欲移至第三步時，需將(2,3,6)的資訊 push 到 stack，(2,3,6)表示第二步的位置在 i=2，j=3，已經偵測的方向為 6 (即 K6)。

接著，從在前往第四步：

1	4	0
0	0	2
3	0	0

目前處於第四步，此時的 stack 內容如下：

3,1,1
2,3,6
1,1,3

撰寫此程式，除了需要有一個 stack 外，還需要一個 n×n 的陣列 (array)。剛開始，陣列所有元素均設定為 0，此時的 0 表示該位置尚未被拜訪過。如果騎士走了三步，則依序在其拜訪過的位置記錄為 1、

2、3。將來若發現此條路徑不通需要退回時，則需將這些位置重設為 0，表示日後騎士尚可由其他路徑拜訪該位置。例如，從第三步退回至第二步時，亦即從位置(3,1)退回至位置(2,3)時，需將位置(3,1)重設為 0，且需從 stack 取出(2,3,6)；代表需退回位置(2,3)，並且上次已經嘗試方向 6 (即 K6)；下次應從方向 7 (即 K6) 嘗試。

輸出：

以左上角(1,1)為起點，分別印出 n=1、2、3、4、5、6 之情形。

如果無解，印出"no solution"。如果有解，則在各個格子內填入拜訪的順序 m，代表該位置是於第 m 步被拜訪的。例如 n=5，可印出：

1	20	17	12	3
16	11	2	7	18
21	24	19	4	13
10	15	6	23	8
25	22	9	14	5

對於有解的 n，印出任一組解即可，不必印出所有的解。

**注意：本習題不得使用 recursive 方法撰寫程式，必須使用 C++ 自己實作 stack 物件，然後以自己所實作的 stack 物件進行 push、pop 等運算。**