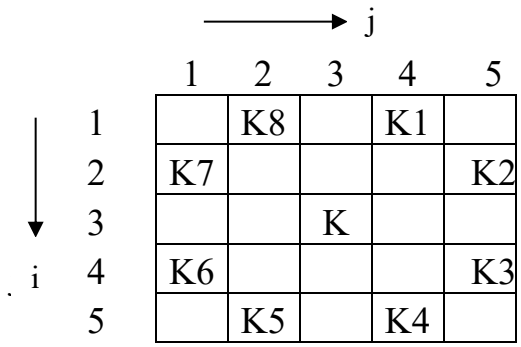
Assignment #3 (An application of stacks)

Goal：Test if a knight can visit each positon of a chess board

Guide：

On an n×n chess board, a knight is initially placed on some position (x,y). You are asked to write a program to report the following (n^2-1) moves of this knight, so that each position on the chess board is visited exactly once. You should complete this assignment with a stack, not recursion. The movement of a knight on a chess board is shown as follows:



Suppose the knight stands on K, then K1 to K8 are eight possible moves of this knight. Assume K is on position (i,j), then the locations of K1 to K8 can be written as follows.

	Difference to i	Difference to j	Location
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)

You can solve this problem by trial and error (backtracking). When the knight is standing on some position K, you can test all possible moves K1 to K8 one by one. If all possible moves fail, then return the previous location from which the knight moved to K, and keep testing other possible moves. Since the knight keep going forward and backward, you

can use a stack to record the path of the knight. Each element in the stack should keep at least three values, which are i, j, and the direction that has been tested.

Take n=3 as an example:

initial:

0	0	0
0	0	0
0	0	0

3 steps after starting from (1,1)

1	0	0
0	0	2
3	0	0

The labels ‘2’ and ‘3’ represent the locations that the knight visit by its second and third move, respectively, and the label ‘0’ means the location has not been visited yet. The stack for the above example is shown as follows:

2,3,6
1,1,3

Contents of the stack: Each record represents the visited state of a chess board. When you want to move from the second position to the third position, you have to push (2,3,6) into stack, where (2,3,6) means that i=2, j=3, and the direction K6, respectively.

Next, the fourth step in the visit:

1	4	0
0	0	2
3	0	0

The stack for the above example is shown as follows:

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

To implement this program, you also need an n×n array, whose elements are initially set to zero. The label ‘0’ means that the location has not been visited yet. For a knight that has moved 3 steps, each visited locations are labeled with 1, 2, 3, which record the order of the

visit. When a knight returns to its previous location, these positions should be reset to 0, so that the knight can still visit these locations from other paths. As an example, suppose that you backtrack from the third move to the second move. That is, you backtrack from the position (3,1) to the position (2,3). The position (3,1) has to be set to zero, and (2,3,6) has to be removed from the stack. It means that the direction “K6” has been tested already. Next, you should test the direction “K7”.

Output：

Set (1,1) as the starting point of the knight. You have to print the path for n=1, 2, 3, 4, 5, and 6. If there is no solution, then print "no solution". If there is a solution, then print the visit order m on the chess board, where m represent that the knight visits the position at step m. Taking n=5 for example, the output is shown as follows:

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

If there is more than one solution for n, you need print any one of the solutions. You do not have to print all solutions.

Note: You must complete this assignment with stack. A recursive program will not be accepted. You must use C++ to implement the stack object, and then you can use push and pop.