Data Structure Assignment 2

Paper Homework

(Textbook p.101 Exercises 8)

8. A complex-valued matrix X is represented by a pair of matrices < a, b >, where a and b contain real values. Write a function that computes the product of tow complex-valued matrices < a, b> and < d, e>, where $< a, b > * < d, e > = (a + ib) * (d + ie) = (ad - be) + i(ae + bd)$. Determine the number of additions and multiplications if the matrices are all $n * n$.

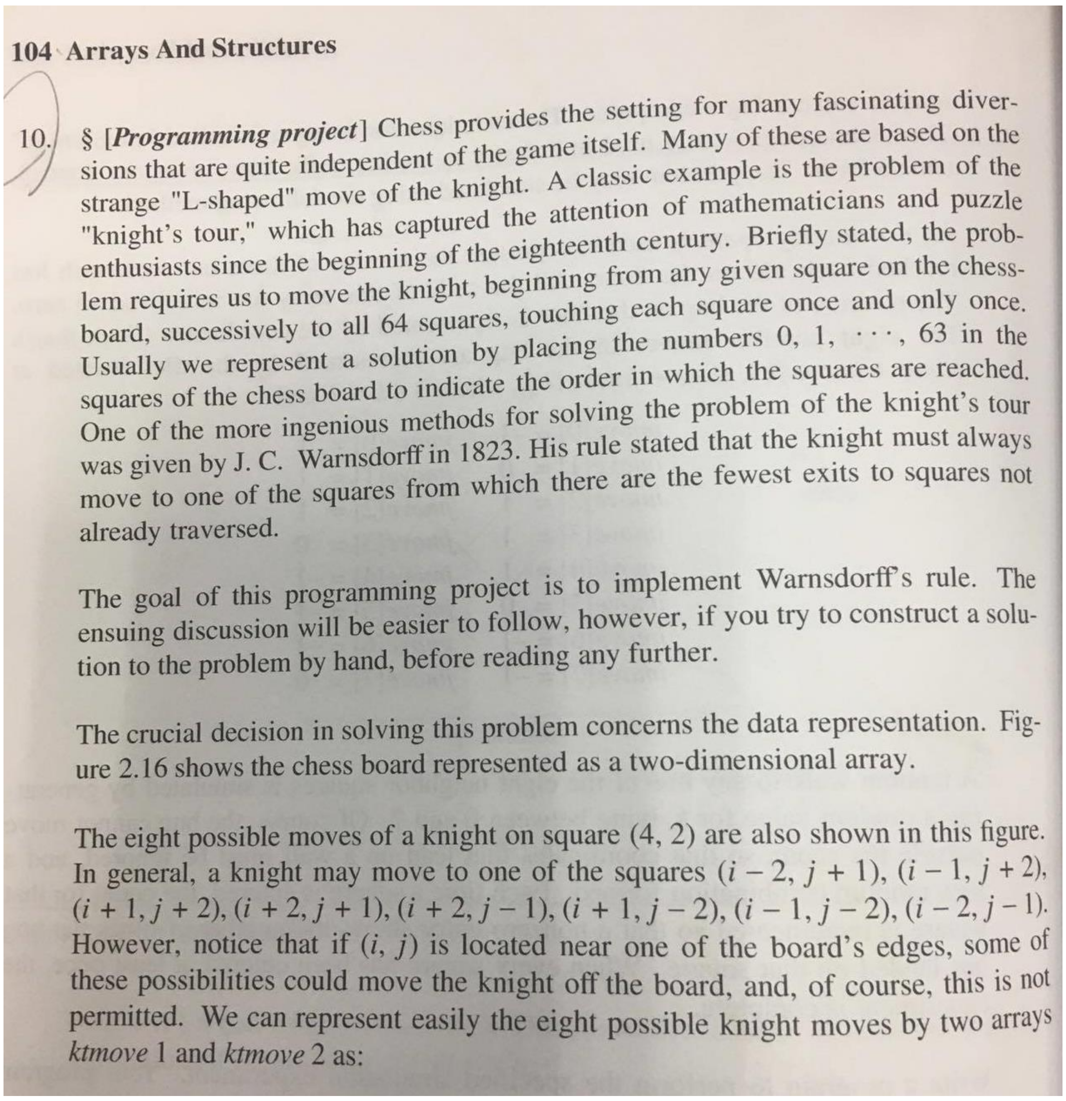
General Information:

- Deadline：2017/10/27 (Please submit to TA in class. We won't accept your homework after class)
- Late homework will not be accepted.
- Please write on A4 papers.
- Notice：You won't get any point if you only write the answer, please list your process and reason.
- Any copies will be scored as zero. Do not plagiarize
- Paper homework TA 張維婷 Email: p76051098@mail.ncku.edu.tw

Data Structure Assignment 2

Programming Homework

(Textbook p.104 Exercises 10)



10. § [***Programming project***] Chess provides the setting for many fascinating diversions that are quite independent of the game itself. Many of these are based on the strange "L-shaped" move of the knight. A classic example is the problem of the "knight's tour," which has captured the attention of mathematicians and puzzle enthusiasts since the beginning of the eighteenth century. Briefly stated, the problem requires us to move the knight, beginning from any given square on the chessboard, successively to all 64 squares, touching each square once and only once. Usually we represent a solution by placing the numbers 0, 1, $\cdots$, 63 in the squares of the chess board to indicate the order in which the squares are reached. One of the more ingenious methods for solving the problem of the knight's tour was given by J. C. Warnsdorff in 1823. His rule stated that the knight must always move to one of the squares from which there are the fewest exits to squares not already traversed.

The goal of this programming project is to implement Warnsdorff's rule. The ensuing discussion will be easier to follow, however, if you try to construct a solution to the problem by hand, before reading any further.

The crucial decision in solving this problem concerns the data representation. Figure 2.16 shows the chess board represented as a two-dimensional array.

The eight possible moves of a knight on square (4, 2) are also shown in this figure. In general, a knight may move to one of the squares $(i-2, j+1)$, $(i-1, j+2)$, $(i+1, j+2)$, $(i+2, j+1)$, $(i+2, j-1)$, $(i+1, j-2)$, $(i-1, j-2)$, $(i-2, j-1)$. However, notice that if (i, j) is located near one of the board's edges, some of these possibilities could move the knight off the board, and, of course, this is not permitted. We can represent easily the eight possible knight moves by two arrays *ktmove* 1 and *ktmove* 2 as:

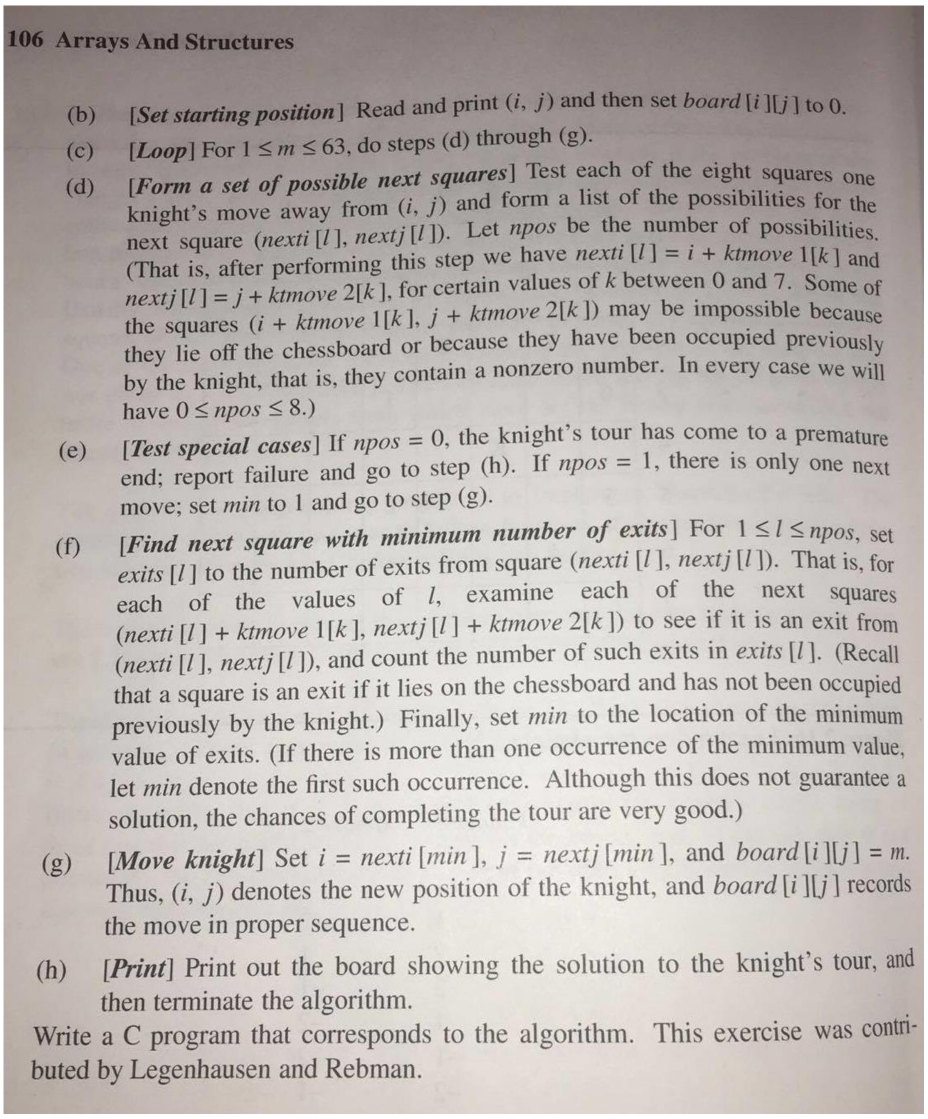
	0	1	2	3	4	5	6	7
0								
1								
2		7		0				
3	6				1			
4			K					
5	5				2			
6		4		3				
7								

Figure 2.16: Legal moves for a knight

ktmove 1	*ktmove* 2
−2	1
−1	2
1	2
2	1
2	−1
1	−2
−1	−2
−2	−1

Then a knight at (i, j) may move to $(i + ktmove1[k], j + ktmove2[k])$, where k is some value between 0 and 7, provided that the new square lies on the chess board. Below is a description of an algorithm for solving the knight's tour problem using Warnsdorff's rule. The data representation discussed in the previous section is assumed.

(a) [***Initialize chessboard***] For $0 \le i, j \le 7$ set *board* $[i][j]$ to 0.



(b) [***Set starting position***] Read and print (i, j) and then set *board* $[i][j]$ to 0.

(c) [***Loop***] For $1 \le m \le 63$, do steps (d) through (g).

(d) [***Form a set of possible next squares***] Test each of the eight squares one knight's move away from (i, j) and form a list of the possibilities for the next square (*nexti* $[l]$, *nextj* $[l]$). Let *npos* be the number of possibilities. (That is, after performing this step we have *nexti* $[l] = i +$ *ktmove* $1[k]$ and *nextj* $[l] = j +$ *ktmove* $2[k]$, for certain values of k between 0 and 7. Some of the squares ($i +$ *ktmove* $1[k]$, $j +$ *ktmove* $2[k]$) may be impossible because they lie off the chessboard or because they have been occupied previously by the knight, that is, they contain a nonzero number. In every case we will have $0 \le npos \le 8$.)

(e) [***Test special cases***] If $npos = 0$, the knight's tour has come to a premature end; report failure and go to step (h). If $npos = 1$, there is only one next move; set *min* to 1 and go to step (g).

(f) [***Find next square with minimum number of exits***] For $1 \le l \le npos$, set *exits* $[l]$ to the number of exits from square (*nexti* $[l]$, *nextj* $[l]$). That is, for each of the values of l, examine each of the next squares (*nexti* $[l]$ + *ktmove* $1[k]$, *nextj* $[l]$ + *ktmove* $2[k]$) to see if it is an exit from (*nexti* $[l]$, *nextj* $[l]$), and count the number of such exits in *exits* $[l]$. (Recall that a square is an exit if it lies on the chessboard and has not been occupied previously by the knight.) Finally, set *min* to the location of the minimum value of exits. (If there is more than one occurrence of the minimum value, let *min* denote the first such occurrence. Although this does not guarantee a solution, the chances of completing the tour are very good.)

(g) [***Move knight***] Set $i =$ *nexti* $[min]$, $j =$ *nextj* $[min]$, and *board* $[i][j] = m$. Thus, (i, j) denotes the new position of the knight, and *board* $[i][j]$ records the move in proper sequence.

(h) [***Print***] Print out the board showing the solution to the knight's tour, and then terminate the algorithm.

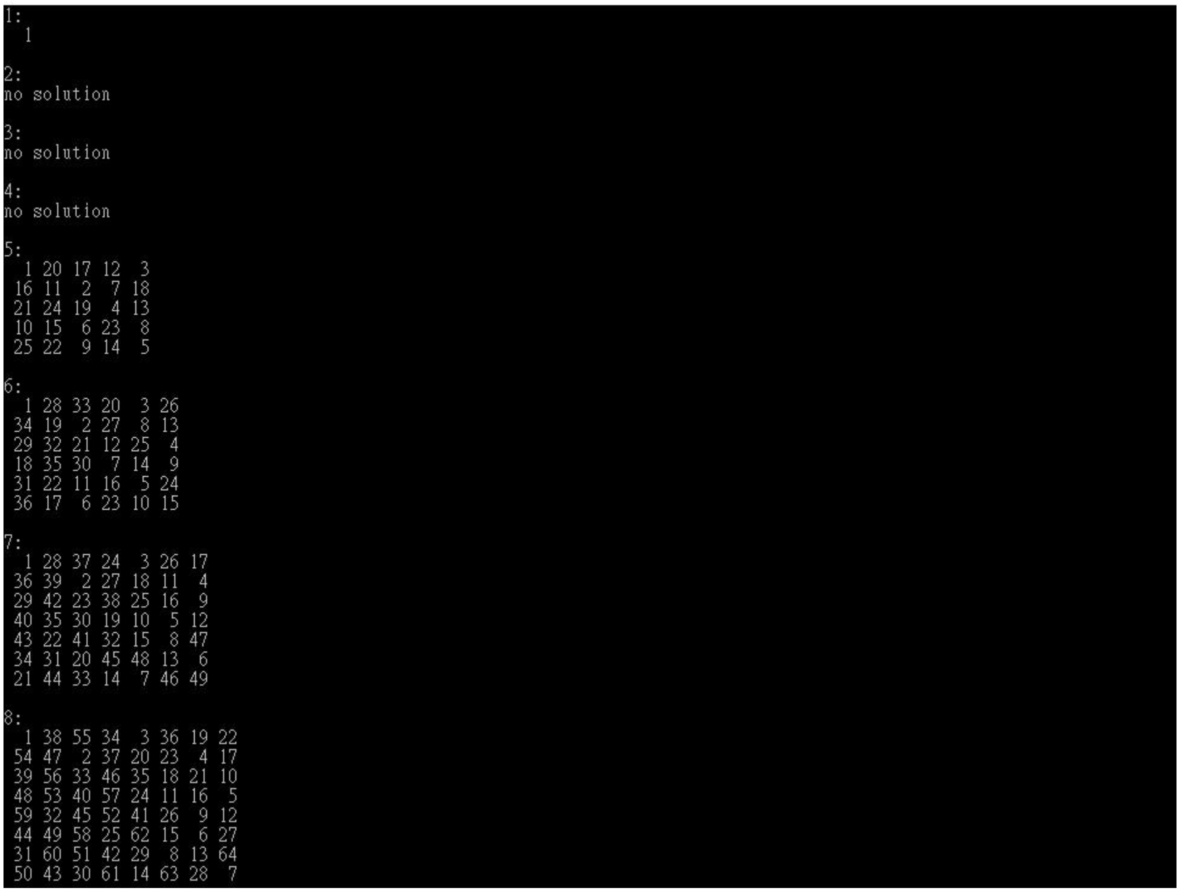
Write a C program that corresponds to the algorithm. This exercise was contributed by Legenhausen and Rebman.

Print out the n*n chess boards (1≦n≦8).

Place the numbers 1 to n^2 in the squares of chess board to indicate the order in which the squares are reached. If no solution, then print out “no solution“.

(1) Set the number 1 at square(0, 0) of chess board.

Output:



```
1:
  1

2:
no solution

3:
no solution

4:
no solution

5:
  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

6:
  1 28 33 20  3 26
 34 19  2 27  8 13
 29 32 21 12 25  4
 18 35 30  7 14  9
 31 22 11 16  5 24
 36 17  6 23 10 15

7:
  1 28 37 24  3 26 17
 36 39  2 27 18 11  4
 29 42 23 38 25 16  9
 40 35 30 19 10  5 12
 43 22 41 32 15  8 47
 34 31 20 45 48 13  6
 21 44 33 14  7 46 49

8:
  1 38 55 34  3 36 19 22
 54 47  2 37 20 23  4 17
 39 56 33 46 35 18 21 10
 48 53 40 57 24 11 16  5
 59 32 45 52 41 26  9 12
 44 49 58 25 62 15  6 27
 31 60 51 42 29  8 13 64
 50 43 30 61 14 63 28  7
```

General Information:

- Deadline：2017/11/03 23:55.
- Upload your assignment to Moodle system.
- Upload file format: Student-Id_Name.rar , Ex.P76991094_王小明.rar
- Your file should consist of the following items: Source Code & Readme file (Program description)
- Late homework will not be accepted.
- Any copies will be scored as zero. Do not plagiarize
- Programming homework TA 傅瑄方 Email: p76051226@mail.ncku.edu.tw