Data Structure: Assignment 2

Due: 2018/11/9 (Friday)

- Requirement: Implementation with C/C++ program
- Total: 100 points (**30 points** for questions 1-2 and **40 points** for question 3)

◆ Submission instructions:

[1]. Write a "**README file"** including the answers to problems 1-3, and a detailed note about the functionality of each of the above programs, and complete instructions on how to run them.

[2]. Make sure you include your name in each program and in the README file. **Each question has one C/C++ program file and one README file.** Make sure all your programs are fully commented, and compile and run correctly on the Linux-based machines.

[3]. Submit your assignment to the portal system by the due date.

1. Write a C/C++ program to convert an infix expression to a postfix expression. In our testing case, we will include the following operators: '(', ')', '+','-','*','\',and '%'. The operators follows the usual precedence. The expression is entered from the standard input.

   Example

   Input format:

   23+7*((3+2)-2)

   Output format:

   23 7 3 2 + 2 - * +

2. Write a C/C++ program to find the next greater element via a **stack data structure**. Given an input, print out the next greater element (which is the first greater element on the right side of current element in array) for every element. If the current element does not have the next greater element, your program will consider next greater element as -1.

   Example

   Input format:

   13, 7, 6, 12

   Output format:

   13 → -1, 7 → 12, 6 → 12, 12 → -1

3. Write a C/C++ program to resolve the knight's tour problem by a stack data structure.

The explanation of problem: A knight's tour is a sequence of moves of a knight on a chessboard such that the knight visits every square of a chessboard exactly once. In this assignment, the knight will start from one of the four corners of the chessboard.

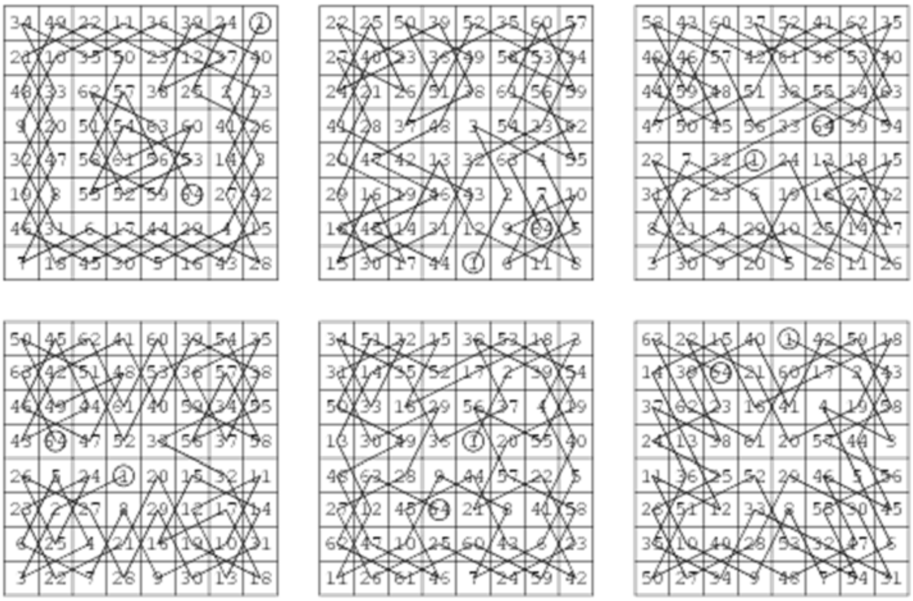

Figure 1. The knight's tour example

Input format:

The size of chessboard: 8 8

The position of knight: 0, 0

Output format:

```
  0  59  38  33  30  17   8  63
 37  34  31  60   9  62  29  16
 58   1  36  39  32  27  18   7
 35  48  41  26  61  10  15  28
 42  57   2  49  40  23   6  19
 47  50  45  54  25  20  11  14
 56  43  52   3  22  13  24   5
 51  46  55  44  53   4  21  12
```

In this example, we create 8-by-8 chessboard and the knight starts from 0, 0.

(**Note that users specify the chessboard size and the beginning location of the knight**)