Chih-Hsiang Wang, 933271081

1.

```
// Language: C
// $ gcc -std=c99 -o knight_tour knight_tour.c
// $ ./knight_tour
// reference: https://www.geeksforgeeks.org/warnsdorffs-algorithm-knights-tour-problem/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// Move pattern on basis of the change of x coordinates and y coordinates respectively
// check next move clockwise from {2, -1}
static int x_move[8] = {2, 1, -1, -2, -2, -1, 1, 2};
static int y_move[8] = {-1, -2, -2, -1, 1, 2, 2, 1};
int board_size;
int open_num = 0, close_num = 0, notFound_num = 0;

// function restricts the knight to remain within the NxN chessboard
int limits(int x, int y) {
    return ((x >= 0 && y >= 0) && (x < board_size && y < board_size));
}

// Checks whether a square is valid and empty or not
int isempty(int *int_array, int x, int y) {
    return (limits(x, y)) && (int_array[y * board_size + x] < 0);
}

// Returns the number of empty squares adjacent to (x, y)
int getDegree(int *int_array, int x, int y) {
    int count = 0;
    for (int i = 0; i < 8; i++) {
        if (isempty(int_array, (x + x_move[i]), (y + y_move[i])))
            count++;
    }

    return count;
}
```

```
// Picks next point using Warnsdorff's heuristic.
// Returns false if it is not possible to pick next point.
int nextMove(int *int_array, int *x, int *y) {
    int min_deg_idx = -1, degree, min_deg = 8, x_next, y_next;
    // Try all board_size adjacent of (*x, *y) starting from a random adjacent.
    // Find the adjacent with minimum degree.
    int start = 0;
    for (int count = 0; count < 8; count++) {
        int i = (start + count) % 8;
        x_next = *x + x_move[i];
        y_next = *y + y_move[i];

        degree = getDegree(int_array, x_next, y_next);
        // print checking path
        if ((isempty(int_array, x_next, y_next)) && degree <= min_deg) {
            min_deg_idx = i;
            min_deg = degree;
        }
    }
    // IF we could not find a next cell
    if (min_deg_idx == -1)
        return 0;

    // Store coordinates of next point
    x_next = *x + x_move[min_deg_idx];
    y_next = *y + y_move[min_deg_idx];

    // Mark next move
    int_array[y_next * board_size + x_next] = int_array[(*y) * board_size + (*x)]
+ 1;

    // Update next point
    *x = x_next;
    *y = y_next;

    return 1;
}

// displays the chessboard with all the legal knight's moves
```

```
void print_board(int *int_array) {
    for (int j = 0; j < board_size; j++) {
        for (int i = 0; i < board_size; i++)
            printf("%d\t", int_array[j * board_size + i]);
        printf("\n");
    }
    printf("\n");
}

// checks its neighbouring sqaures, if the knight ends on a square that is
one
// knight's move from the beginning square, then tour is closed
int neighbour(int x, int y, int init_x, int init_y) {
    for (int i = 0; i < board_size; i++) {
        if (((x + x_move[i]) == init_x) && ((y + y_move[i]) == init_y))
            return 1;
    }

    return 0;
}

// Generates the legal moves using warnsdorff's heuristics. Returns false if
not possible
int getTour(int *int_array, int decide_row, int decide_col) {
    // Filling up the chessboard matrix with -1's int_array[n * n];
    for (int i = 0; i < board_size * board_size; i++)
        int_array[i] = -1;

    // Random initial position || base on the input to decide initial position
    int init_x = decide_row == -1 ? rand() % board_size : decide_col;
    int init_y = decide_col == -1 ? rand() % board_size : decide_row;

    // Current points are the same as initial points
    int x = init_x, y = init_y;
    int_array[y * board_size + x] = 1; // Mark first move.

    // Keep picking next points using Warnsdorff's heuristic
    for (int i = 0; i < board_size * board_size - 1; i++) {
        if (nextMove(int_array, &x, &y) == 0) {
            notFound_num++;
```

```
            printf("\n<<can not find the tour>>\n\n");
            print_board(int_array);
            return 0;
        }
    }

    // Check if tour is closed (Can end at starting point)
    if (!neighbour(x, y, init_x, init_y)) {
        open_num++;
        printf("\n<<open tour>>\n\n");
    }
    else {
        close_num++;
        printf("\n<<closed tour>>\n\n");
    }
    print_board(int_array);
    return 1;
}

// Nomrmal process running for one time
int main() {
    printf("\nkey in the n for the n*n board\n");
    scanf("%d", &board_size);

    int *int_array = malloc(sizeof(int) * board_size * board_size);

    // To make sure that different random initial positions are picked.
    srand(time(NULL));

    getTour(int_array, -1, -1);

    return 0;
}
```

2.

```
key in the n for the n*n board
5

<<open tour>>

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


<<can not find the tour>>

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


<<open tour>>

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


<<can not find the tour>>

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


<<open tour>>

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


<<can not find the tour>>

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


<<open tour>>

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


<<can not find the tour>>

2       -1      -1      -1      8
-1      -1      1       -1      -1
-1      3       -1      7       -1
-1      -1      5       -1      -1
4       -1      -1      -1      6


<<open tour>>

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


<<can not find the tour>>

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

<<can not find the tour>>

11	-1	5	-1	17
6	-1	10	-1	4
1	12	-1	16	9
-1	7	14	3	-1
13	2	-1	8	15

<<can not find the tour>>

8	-1	-1	-1	6
-1	-1	7	-1	-1
-1	1	-1	5	-1
-1	-1	3	-1	-1
2	-1	-1	-1	4

<<open tour>>

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

<<can not find the tour>>

6	-1	-1	-1	8
-1	-1	7	-1	-1
-1	5	-1	1	-1
-1	-1	3	-1	-1
4	-1	-1	-1	2

<<open tour>>

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

<<can not find the tour>>

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

<<open tour>>

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

<<can not find the tour>>

6	-1	-1	-1	4
-1	-1	5	-1	-1
-1	7	-1	3	-1
-1	-1	1	-1	-1
8	-1	-1	-1	2

<<open tour>>

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

<<can not find the tour>>

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

```
<<open tour>>

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


<<can not find the tour>>

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


<<open tour>>

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


<<can not find the tour>>

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


<<open tour>>

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

open tour: 12
close tour: 0
not found: 13
```

3.
==========
// check next move clockwise from {2, -1}

My program does not always find a tour in 5x5 board.

The tours I can find in 5x5 board is only open tour.
There are two different consequences, open tour & cannot find the tour.

open tour: 12
close tour: 0
not found: 13

So there are 12 different tours my program can find.
==========

4.

```
key in the n for the n*n board
6

<<open tour>>

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


key in the n for the n*n board
6

<<open tour>>

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


key in the n for the n*n board
6

<<open tour>>

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

```
key in the n for the n*n board
6

<<open tour>>

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

==========

From 4 different initial square, I found 4 open tour.
However, my program can also find closed tour and cannot find the tour.

==========

```
<<closed tour>>

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


<<can not find the tour>>

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

==========
// check next move clockwise from {2, -1}

My program does not always find a tour in 6x6 board.

The tours I can find in 6x6 board is open tour & close tour.
There are three different consequences,
open tour & close tour & cannot find the tour.

open tour: 31
close tour: 4
not found: 1

So there are 31+4=35 different tours my program can find.
==========

5.

```
key in the n for the n*n board
8

<<open tour>>

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

```
key in the n for the n*n board
8

<<open tour>>

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

```
key in the n for the n*n board
8

<<open tour>>

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

```
key in the n for the n*n board
8

<<open tour>>

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

==========
From 4 different initial square, I found 4 open tour.
However, my program can also find closed tour and cannot find the tour.
==========

```
<<closed tour>>

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

<<can not find the tour>>

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

==========
// check next move clockwise from {2, -1}

My program does not always find a tour in 8x8 board.

The tours I can find in 8x8 board is open tour & close tour.
There are three different consequences,
open tour & close tour & cannot find the tour.

open tour: 59
close tour: 4
not found: 1

So there are 59+4=63 different tours my program can find.
==========