933293709
Jui-Hung Lu
Reference: https://www.geeksforgeeks.org/warnsdorffs-algorithm-knights-tour-problem/

1.

```
// C program
//gcc -o main main.c
#include <stdio.h>
#include <stdbool.h>
#include <time.h>
#include <stdlib.h>
static int cx[8] = {1,1,2,2,-1,-1,-2,-2};
static int cy[8] = {2,-2,1,-1,2,-2,1,-1};
int N;

bool limits(int x, int y)
{
   return ((x >= 0 && y >= 0) && (x < N && y < N));
}

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

/* Returns the number of empty squares adjacent to (x, y) */
int nextMovenum(int a[], int x, int y)
{
   int count = 0;
   for (int i = 0; i < 8; ++i)
      if (isempty(a, (x + cx[i]), (y + cy[i])))
         count++;

   return count;
}

// Picks next point using Warnsdorff's heuristic.
// Returns false if it is not possible to pick next point.
bool nextMove(int a[], int *x, int *y)
{
   int min_deg_idx = -1, c, min_deg = (8+1), nx, ny;

   // Try all adjacent of (*x, *y), find the adjacent with minimum degree.
```

```
   int start = 0;
   for (int count = 0; count < 8; ++count)
   {
      int i = (start + count)%8;
      nx = *x + cx[i];
      ny = *y + cy[i];
      if ((isempty(a, nx, ny)) &&
         (c = nextMovenum(a, nx, ny)) < min_deg)
      {
         min_deg_idx = i;
         min_deg = c;
      }
   }

   // IF we could not find a next cell
   if (min_deg_idx == -1)
      return false;

   // Store coordinates of next point
   nx = *x + cx[min_deg_idx];
   ny = *y + cy[min_deg_idx];

   // Mark next move
   a[ny*N + nx] = a[(*y)*N + (*x)]+1;

   // Update next point
   *x = nx;
   *y = ny;

   return true;
}
/* checks the tour is closed if return ture*/
bool checkclosed(int x, int y, int xx, int yy)
{
   for (int i = 0; i < N; ++i)
      if (((x+cx[i]) == xx)&&((y + cy[i]) == yy))
         return true;

   return false;
}
int main()
{
   // To make sure that different random
   // initial positions are picked.
```

```
    srand(time(NULL));
    printf("enter a number\n");
    scanf("%d",&N);
    int a[N*N];
    for (int i = 0; i< N*N; ++i)
        a[i] = -1;

    // Randome initial position
    int sx = rand()%N;
    int sy = rand()%N;
    int x = sx, y = sy;
    // Mark first move.
    a[y*N+x] = 1;

    // Keep picking next points using Warnsdorff's heuristic
    for (int i = 0; i < N*N-1; ++i)
        if (nextMove(a, &x, &y) == 0){
            printf("\n<<can not find the tour>>\n");
            for (int i = 0; i < N; ++i)
            {
                for (int j = 0; j < N; ++j)
                    printf("%d\t",a[j*N+i]);
                printf("\n");
            }
            exit(0);
        }
    // Check if tour is closed
    if (!checkclosed(x, y, sx, sy)){
        printf("opened solution\n");
    }else{
        printf("closed solution\n");
    }
    //print
    for (int i = 0; i < N; ++i)
    {
        for (int j = 0; j < N; ++j)
            printf("%d\t",a[j*N+i]);
        printf("\n");
    }
    return 0;
}
```

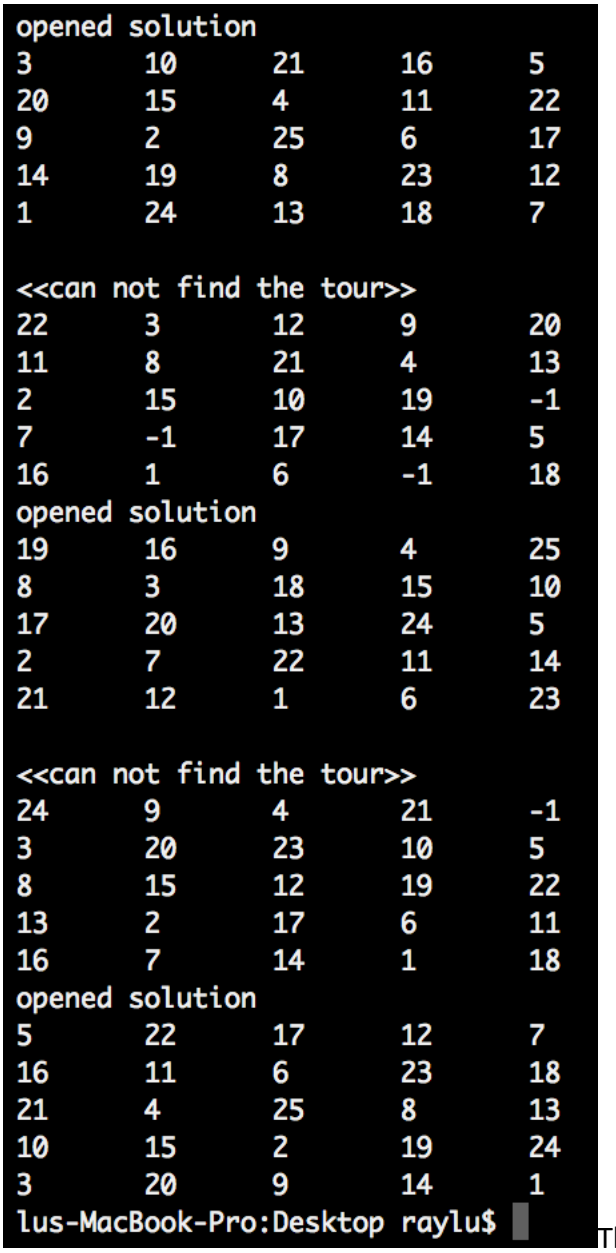
2.

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

<<can not find the tour>>
16      1       6       13      24
7       12      15      20      5
2       17      10      23      14
11      8       19      4       21
18      3       22      9       -1

<<can not find the tour>>
13      -1      1       6       11
2       7       12      -1      -1
-1      14      -1      10      5
8       3       16      -1      -1
15      -1      9       4       17

<<can not find the tour>>
22      7       12      1       20
11      2       21      6       13
8       15      10      19      -1
3       -1      17      14      5
16      9       4       -1      18
opened solution
7       16      11      22      1
10      21      8       17      12
15      6       23      2       25
20      9       4       13      18
5       14      19      24      3
```

```
<<can not find the tour>>
-1      7       2       13      16
1       14      17      8       3
6       -1      12      15      20
11      18      21      4       9
-1      5       10      19      22
opened solution
25      6       17      12      23
16      1       24      5       10
7       18      11      22      13
2       15      20      9       4
19      8       3       14      21

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

<<can not find the tour>>
22      5       14      11      20
13      10      21      6       1
4       15      12      19      -1
9       -1      17      2       7
16      3       8       -1      18
```

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

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

<<can not find the tour>>
4       -1      -1      -1      2
-1      -1      3       -1      -1
-1      5       -1      1       -1
-1      -1      7       -1      -1
6       -1      -1      -1      8
opened solution
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
```

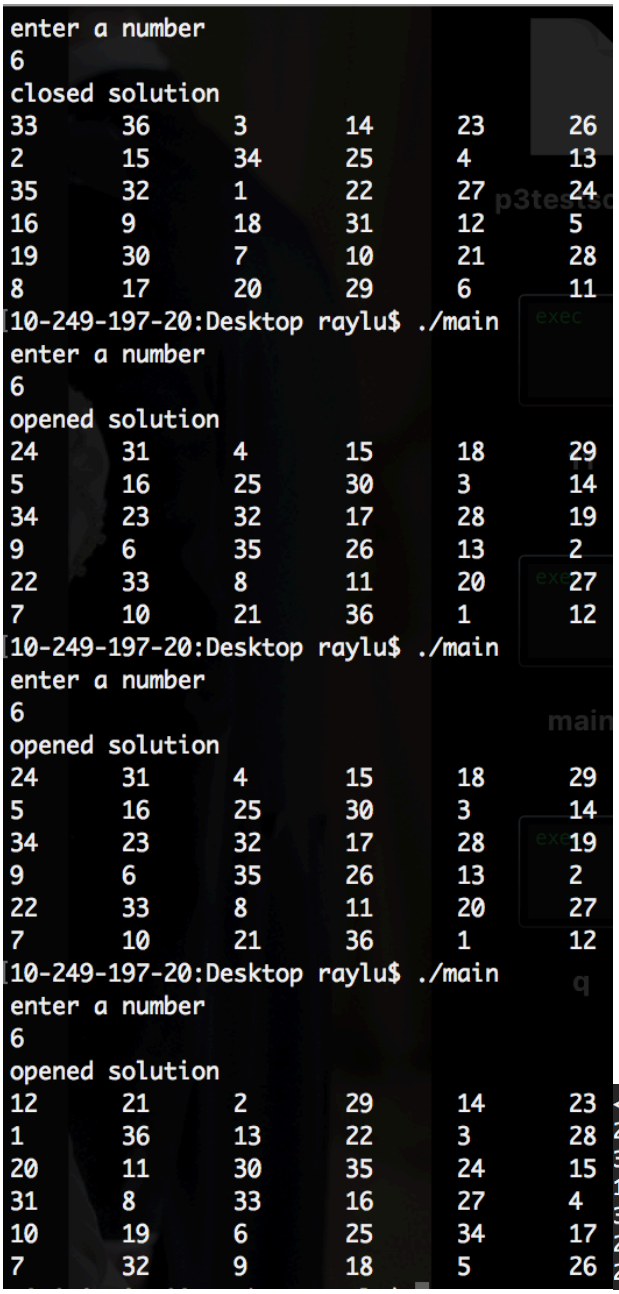
```
<<can not find the tour>>
16      13      8       3       24
7       2       15      20      9
14      17      12      23      4
1       6       19      10      21
18      11      22      5       -1
opened solution
21      8       3       16      19
2       17      20      9       4
7       22      13      18      15
12      1       24      5       10
23      6       11      14      25

<<can not find the tour>>
4       -1      -1      -1      6
-1      -1      5       -1      -1
-1      3       -1      7       -1
-1      -1      1       -1      -1
2       -1      -1      -1      8
opened solution
21      4       13      10      23
14      9       22      5       12
3       20      11      24      17
8       15      18      1       6
19      2       7       16      25

<<can not find the tour>>
22      19      10      5       -1
9       4       21      18      11
20      15      12      -1      6
3       8       17      14      1
16      13      2       7       -1
```

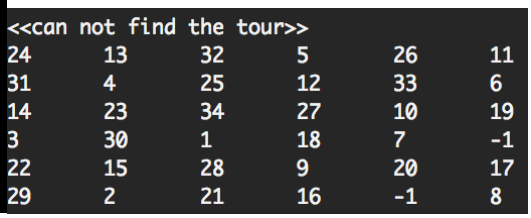


There are 12 opened,0 closed and 13 no solution

3.
According to the question2, this program cannot always find the tour.
Besides, after testing several times, this program cannot find a closed solution for 5*5 game board.
There are only two main kinds of solution for 5*5 game board, opened solution and “cannot find a solution”.
I have tried to print only closed, if it didn’t find a closed solution, it would keep running.
However, it runs few minutes and still running and didn’t print any closed solution.
Therefore, I guess there is no closed solution for 5*5 knight tour.
For opened solution, there are 12 kinds of different solutions.
12 opened solution
0 closed solution
13 no solution

4.





According to the solution, this program can find a solution for 6*6 game board. It can find both closed and opened solution, and no solution. Most of the solution is opened solution.

There are three main kinds of solution for 6*6 game board, opened, closed and no solution.

For opened and closed solution, there are 35 kinds of different solutions.

31 opened solution

4 closed solution

1 no solution

5.

```
opened solution
23      26      63      6       45      28      47      8
62      5       24      27      58      7       44      29
25      22      37      64      41      46      9       48
4       61      40      57      38      59      30      43
21      36      19      60      55      42      49      10
18      3       56      39      50      13      54      31
35      20      1       16      33      52      11      14
2       17      34      51      12      15      32      53
[10-249-197-20:Desktop raylu$ ./main
enter a number
8
opened solution
34      9       56      29      36      11      38      27
63      30      35      10      57      28      15      12
8       33      62      55      14      37      26      39
51      64      31      44      61      58      13      16
32      7       52      59      54      1       40      25
47      50      45      22      43      60      17      2
6       21      48      53      4       19      24      41
49      46      5       20      23      42      3       18
[10-249-197-20:Desktop raylu$ ./main
enter a number
8
closed solution
20      5       30      33      18      3       38      41
29      32      19      4       43      40      17      2
6       21      56      31      34      37      42      39
55      28      35      44      57      50      1       16
22      7       60      51      36      45      58      49
27      54      25      46      59      64      15      12
8       23      52      61      10      13      48      63
53      26      9       24      47      62      11      14
[10-249-197-20:Desktop raylu$ ./main
enter a number
8
opened solution
48      1       18      23      52      27      16      25
19      22      49      56      17      24      53      28
2       47      20      51      54      61      26      15
21      50      55      62      57      36      29      60
46      3       44      37      64      59      14      35
9       6       63      58      43      32      39      30
4       45      8       11      38      41      34      13
7       10      5       42      33      12      31      40
<<can not find the tour>>
20      39      16      51      56      41      14      45
17      2       19      40      15      44      55      42
38      21      50      1       52      57      46      13
3       18      37      58      47      54      43      60
22      49      24      53      36      59      12      -1
7       4       27      48      25      -1      35      32
28      23      6       9       30      33      -1      11
5       8       29      26      -1      10      31      34
```

According to the solution, this program can find a solution for 8*8 game board.
It can find both closed and opened solution, and no solution. Most of the solution is opened solution.
There are three main kinds of solution for 8*8 game board, opened, closed and no solution.
For opened and closed solution, there are 63 kinds of different solutions.
59 opened solution
4 closed solution
1 no solution