|  |  |  |
| --- | --- | --- |
| 姓名：李昱昇 | 測驗日期：3/21 | 分數： |

※請於**面試前一天完成**測驗，並將檔案回傳至resume@onelab.tw，謝謝。

【題　　目】：

Write a function that returns all distinct solutions to the 8-queens puzzle.

Each solution contains a distinct board configuration of the 8-queens' placement, where 'Q' and '.' both indicate a queen and an empty space respectively.

請編寫一個函數，返回8皇后難題的所有不同答案。

每個答案均包含一個獨特的8皇后在棋盤上的配置，其中 'Q' 及 '.' 分別代表一個皇后和一個空白。

---------------------

Example solution for 4-queen puzzle

【答題範例】：四皇后的答案範例

Output:

// Solution 1

.Q..

...Q

Q...

..Q.

// Solution 2

..Q.

Q...

...Q

.Q..

【開始解題】：

static void Main(string[] args)

        {  Queen.QueenFun(8);

            Console.Read();

        }

class Queen

    {

        static List<int> queenX = new List<int>();

        static List<int> queenY = new List<int>();

        static int succCount = 0;

        public static void QueenFun(int n)

        {

            int rj = 1;

            for(int i = queenX.Count + 1; i <= n; i++)

            {

                for(int j = rj; j <= n; j++)

                {

                    if (CheckQueen(i, j, queenX, queenY))

                    {

                        queenX.Add(i);

                        queenY.Add(j);

                        rj = 1;

                        if (queenX.Count==n && queenY.Count == n)

                        {

                            InsertSucc();

                            rj = queenY[i-1]+1;

                            if (rj > n)

                            {

                                int temp = queenY[i - 2] + 1;

                                queenX.RemoveAt(i - 1);

                                queenY.RemoveAt(i - 1);

                                queenX.RemoveAt(i - 2);

                                queenY.RemoveAt(i - 2);

                                i -= 2;

                                rj = temp;

                            }

                            else

                            {

                                queenX.RemoveAt(i - 1);

                                queenY.RemoveAt(i - 1);

                                i--;

                            }

                        }

                        break;

                    }

                    if (j == n)

                    {

                        if (i < 2)

                        {

                            i = n + 1;

                            break;

                        }

                        rj = queenY[i - 2] + 1;

                        if (rj > n)

                        {

                            if (i < 3)

                            {

                                i = n + 1;

                                break;

                            }

                            int temp = queenY[i - 3] + 1;

                            queenX.RemoveAt(i - 2);

                            queenY.RemoveAt(i - 2);

                            queenX.RemoveAt(i - 3);

                            queenY.RemoveAt(i - 3);

                            i = queenX.Count;

                            rj = temp;

                        }

                        else

                        {

                            queenX.RemoveAt(i - 2);

                            queenY.RemoveAt(i - 2);

                            i -= 2;

                        }

                        break;

                    }

                }

            }

        }

        public static bool CheckQueen(int putX,int putY,List<int> x,List<int> y)

        {

           for(int i = 0; i < x.Count; i++)

            {

                if (putX == x[i])

                    return false;

                if (putY == y[i])

                    return false;

                if (putX - putY == x[i] - y[i])

                    return false;

                if (putX + putY == x[i] + y[i])

                    return false;

            }

            return true;

        }

        static void InsertSucc()

        {

            string temp = "";

            string result = "";

            succCount++;

            Console.WriteLine("Solution: "+succCount);

            for(int i = 0; i < queenX.Count; i++)

            {

                for (int y = 1; y <= queenY.Count - 1; y++)

                    temp += ".";

                result = temp.Insert(queenY[i] - 1, "Q");

                Console.WriteLine(result);

                temp = "";

            }

        }

    }