using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Zadacha14

{

    class Program

    {

        static void Main(string[] args)

        {

            var n = int.Parse(Console.ReadLine());

            char[,] bakerySquare = new char[n, n];

            FillMatrix(n, bakerySquare);

            int sum = 0;

            char startPosition = bakerySquare[0, 0];

            int indexRow = 0, indexCol = 0;

            StartPosition(n, bakerySquare, ref startPosition, ref indexRow, ref indexCol);

            bakerySquare[indexRow, indexCol] = '-';

            while (true)

            {

                string command = Console.ReadLine();

                switch (command)

                {

                    case "up":

                        {

                            indexRow--;

                            if (!IsValid(indexRow, indexCol, n)) break;

                            Pillars(n, bakerySquare, ref indexRow, ref indexCol);

                            if (char.IsDigit(bakerySquare[indexRow, indexCol]))

                                sum += (int)bakerySquare[indexRow, indexCol] - 48;

                            bakerySquare[indexRow, indexCol] = '-';

                            break;

                        }

                    case "down":

                        {

                            indexRow++;

                            if (!IsValid(indexRow, indexCol, n)) break;

                            Pillars(n, bakerySquare, ref indexRow, ref indexCol);

                            if (char.IsDigit(bakerySquare[indexRow, indexCol]))

                                sum += (int)bakerySquare[indexRow, indexCol] - 48;

                            bakerySquare[indexRow, indexCol] = '-';

                            break;

                        }

                    case "left":

                        {

                            indexCol--;

                            if (!IsValid(indexRow, indexCol, n)) break;

                            Pillars(n, bakerySquare, ref indexRow, ref indexCol);

                            if (char.IsDigit(bakerySquare[indexRow, indexCol]))

                                sum += (int)bakerySquare[indexRow, indexCol] - 48;

                            bakerySquare[indexRow, indexCol] = '-';

                            break;

                        }

                    case "right":

                        {

                            indexCol++;

                            if (!IsValid(indexRow, indexCol, n)) break;

                            Pillars(n, bakerySquare, ref indexRow, ref indexCol);

                            if (char.IsDigit(bakerySquare[indexRow, indexCol]))

                                sum += (int)bakerySquare[indexRow, indexCol] - 48;

                            bakerySquare[indexRow, indexCol] = '-';

                            break;

                        }

                }

                if (!IsValid(indexRow, indexCol, n)) break;

                if (sum >= 50)

                {

                    bakerySquare[indexRow, indexCol] = 'S'; break;

                }

            }

            if (sum >= 50)

                Console.WriteLine("Good news! You succeeded in collecting enough money!");

            else Console.WriteLine("Bad news, you are out of the bakery.");

            Console.WriteLine($"Money: {sum}");

            PrintMatrix(n, bakerySquare);

        }

        private static void Pillars(int n, char[,] bakerySquare, ref int indexRow, ref int indexCol)

        {

            if (bakerySquare[indexRow, indexCol] == 'O')

            {

                bakerySquare[indexRow, indexCol] = '-';

                for (int i = 0; i < n; i++)

                    for (int j = 0; j < n; j++)

                    {

                        if (bakerySquare[i, j] == 'O')

                        {

                            indexRow = i;

                            indexCol = j;

                            bakerySquare[indexRow, indexCol] = '-';

                        }

                    }

            }

        }

        private static bool IsValid(int indexRow, int indexCol, int n)

        {

            if (indexRow < 0 || indexRow > n - 1 || indexCol < 0 || indexCol > n - 1)

                return false;

            return true;

        }

        private static void StartPosition(int n, char[,] bakerySquare, ref char startPosition, ref int indexRow, ref int indexCol)

        {

            for (int i = 0; i < n; i++)

            {

                for (int j = 0; j < n; j++)

                {

                    if (bakerySquare[i, j] == 'S')

                    {

                        startPosition = bakerySquare[i, j];

                        indexRow = i; indexCol = j;

                        break;

                    }

                }

                if (startPosition != bakerySquare[0, 0]) break;

            }

        }

        private static void PrintMatrix(int n, char[,] bakerySquare)

        {

            for (int i = 0; i < n; i++)

            {

                for (int j = 0; j < n; j++)

                {

                    Console.Write(bakerySquare[i, j]);

                }

                Console.WriteLine();

            }

        }

        private static void FillMatrix(int n, char[,] bakerySquare)

        {

            for (int i = 0; i < n; i++)

            {

                char[] line = Console.ReadLine().ToCharArray();

                for (int j = 0; j < n; j++)

                {

                    bakerySquare[i, j] = line[j];

                }

            }

        }

    }

}