using System;

using System.Linq;

namespace \_02.Bee

{

internal class Program

{

static void Main(string[] args)

{

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

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

FillMatrix(n, garden);

int sum = 0;

char startPosition = garden[0, 0];

int indexRow = 0, indexCol = 0;

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

garden[indexRow, indexCol] = '.';

string command = "";

while ((command = Console.ReadLine()) != "End")

{

switch (command)

{

case "up":

{

indexRow--;

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

if (garden[indexRow, indexCol] == 'f')

sum++;

else if (garden[indexRow, indexCol] == 'O')

{

garden[indexRow, indexCol] = '.';

indexRow--;

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

if (garden[indexRow, indexCol] == 'f')

sum++;

}

garden[indexRow, indexCol] = '.';

break;

}

case "down":

{

indexRow++;

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

if (garden[indexRow, indexCol] == 'f')

sum++;

else if (garden[indexRow, indexCol] == 'O')

{

garden[indexRow, indexCol] = '.';

indexRow++;

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

if (garden[indexRow, indexCol] == 'f')

sum++;

}

garden[indexRow, indexCol] = '.';

break;

}

case "left":

{

indexCol--;

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

if (garden[indexRow, indexCol] == 'f')

sum++;

else if (garden[indexRow, indexCol] == 'O')

{

garden[indexRow, indexCol] = '.';

indexCol--;

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

if (garden[indexRow, indexCol] == 'f')

sum++;

}

garden[indexRow, indexCol] = '.';

break;

}

case "right":

{

indexCol++;

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

if (garden[indexRow, indexCol] == 'f')

sum++;

else if (garden[indexRow, indexCol] == 'O')

{

garden[indexRow, indexCol] = '.';

indexCol++;

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

if (garden[indexRow, indexCol] == 'f')

sum++;

}

garden[indexRow, indexCol] = '.';

break;

}

}

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

}

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

{

Console.WriteLine("The bee got lost!");

}

else

{

garden[indexRow, indexCol] = 'B';

}

if (sum > 4)

{

Console.WriteLine($"Great job, the bee managed to pollinate {sum} flowers!");

}

else

{

Console.WriteLine($"The bee couldn't pollinate the flowers, she needed {5 - sum} flowers more");

}

PrintMatrix(n, garden);

}

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

{

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

return false;

return true;

}

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

{

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

{

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

{

if (garden[i, j] == 'B')

{

startPosition = garden[i, j];

indexRow = i; indexCol = j;

break;

}

}

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

}

}

static void PrintMatrix(int n, char[,] garden)

{

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

{

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

{

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

}

Console.WriteLine();

}

}

static void FillMatrix(int n, char[,] garden)

{

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

{

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

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

{

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

}

}

}

}

}