Author

[curious\_db97](https://codefights.com/profile/curious_db97)

https://codefights.com/img/coins_new.png2000



Who doesn't want 2000 Coins, Huh?

You being a smartass coder, immediately saw that going "NORTH" and then "SOUTH" is not reasonable, better stay put! Similarly going "EAST" and then "WEST" is a waste of time. Hence, the directions given in the poster can be simplified in two steps as shown below:

* Step 1: Remove the following pair of opposite directions:
* directions = [ <"NORTH", "SOUTH">,
* "SOUTH", <"EAST", "WEST">,
* "NORTH", <"WEST", "EAST">,
* "NORTH", "WEST", "SOUTH",
* "EAST", <"NORTH", "SOUTH">,
* <"EAST", "WEST">, "NORTH",
* "NORTH", "WEST"]
* Step 2: Remove the following pair of opposite directions:
* directions = [ <"SOUTH", "NORTH">, "NORTH",
* "WEST", "SOUTH", "EAST",
* "NORTH", "NORTH", "WEST"]

The resulting path cannot be further simplified. Hence, the shorter version of the directions given in the poster would be ["NORTH", "WEST", "SOUTH", "EAST", "NORTH", "NORTH", "WEST"].

Your task is to create a simplified version of the givendirections to get the desired 2000 coins.

**Example**

* For
* directions = ["NORTH",
* "SOUTH",
* "SOUTH",
* "EAST",
* "WEST",
* "NORTH",
* "WEST"]

the output should be  
ReduceDirections(directions) = ["WEST"].

* For
* directions = ["NORTH",
* "SOUTH",
* "SOUTH",
* "EAST",
* "WEST",
* "NORTH"

the output should be  
ReduceDirections(directions) = [].

**Input/Output**

* **[time limit] 3000ms (cs)**
* **[input] array.string directions**

An array consisting of directions, where each direction is one of the following: "NORTH","SOUTH", "EAST" or "WEST".

*Constraints:*  
5 ≤ directions.length ≤ 30.

* **[output] array.string**

A simplified version of the given directions.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static string[] eliminarAdjacentes(string[] directions)

{

bool[] marcas = new bool[directions.Length];

for (int i = 0; i + 1 < directions.Length; i++)

{

if (!marcas[i] && !marcas[i + 1])

{

if ((directions[i] == "NORTH" && directions[i + 1] == "SOUTH")

|| (directions[i + 1] == "NORTH" && directions[i] == "SOUTH")

|| (directions[i] == "EAST" && directions[i + 1] == "WEST")

|| (directions[i + 1] == "EAST" && directions[i] == "WEST"))

{

marcas[i] = true;

marcas[i + 1] = true;

}

}

}

List<string> ans = new List<string>();

for (int i = 0; i < marcas.Length; i++)

{

if (!marcas[i])

{

ans.Add(directions[i]);

}

}

return ans.ToArray();

}

static bool esIgual(string[] a, string[] b)

{

for (int i = 0; i < a.Length; i++)

{

if (a[i] != b[i])

{

return false;

}

}

return true;

}

static string[] ReduceDirections(string[] directions)

{

//string[] elim = eliminarAdjacentes(directions);

//while (true)

//{

// if (esIgual(elim, eliminarAdjacentes(elim)))

// {

// break;

// }

// if (elim.Length == 0)

// {

// break;

// }

// elim = eliminarAdjacentes(elim);

//}

string[] elim = eliminarAdjacentes(directions);

if (elim.Length > 0)

{

elim = eliminarAdjacentes(elim);

}

return elim;

}

static void Main(string[] args)

{

string[] directions = {"NORTH",

"SOUTH",

"SOUTH",

"EAST",

"WEST",

"NORTH",

"WEST",

"EAST",

"NORTH",

"WEST",

"SOUTH",

"EAST",

"NORTH",

"SOUTH",

"EAST",

"WEST",

"NORTH",

"NORTH",

"WEST"};

foreach (string s in ReduceDirections(directions))

{

Console.WriteLine(s);

}

Console.ReadLine();

}

}

}