**Only Survivor**

Some guys holding swords are playing a game. This game will be the last one for most of them.

The rules of the game are the following: the players are standing in a circle. Person number 1 starts the game by killing the guy to the left of him. After that, the next living person to his left kills his the closest person to their left of him who is still alive. The game continues until there's only one person left.

Your task is, given the number of players playing the game, to find the last survivor and return the one-based position of that person. Assume that the person who starts the game has number 1, and the players are enumerated in the clockwise direction.

**Example**

For n = 10, the output should be  
onlySurviver(n) = 5.

Here's why:

|  |  |
| --- | --- |
| http://fs5.directupload.net/images/161028/58dbqxly.png | * Person 1 kills Person 2 * Person 3 kills Person 4 * Person 5 kills Person 6 * Person 7 kills Person 8 * Person 9 kills Person 10 * Person 1 kills Person 3 * Person 5 kills Person 7 * Person 9 kills Person 1 * Person 5 kills Person 9   The output should be: 5. |

**Input/Output**

* **[time limit] 3000ms (cs)**
* **[input] integer players**

*Constraints:*  
0 < players < 231.

* **[output] integer**

1-based position of the survivor of the game.

<https://codefights.com/challenge/gmYZRfrxWt6Ka7gYo>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static string toBin(int n)

{

string res = "";

while (n > 0)

{

res = res.Insert(0, (n % 2).ToString());

// res += (n % 2).ToString();

n /= 2;

}

return res;

}

public static int BinarioADecimal(String NumeroBinario)

{

// NumeroBinario = validaBin(NumeroBinario); //llama a la validacion del numero binario

char[] temporal = NumeroBinario.ToCharArray();

Array.Reverse(temporal);

int NumeroDecimal = 0;

for (int i = NumeroBinario.Length - 1; i >= 0; i--)

{

NumeroDecimal += Convert.ToInt32(int.Parse(temporal[i].ToString()) \* Math.Pow(2, i));

}

return NumeroDecimal;

}

static int onlySurviver(int players)

{

string binPlay = toBin(players);

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

int i = 0;

while (i < binPlay.Length)

{

string concat = "";

while (i < binPlay.Length && binPlay[i] == '0')

{

concat += '0';

i++;

}

if (concat.Length > 0)

{

grupos.Add(concat);

}

//concat = "";

while (i < binPlay.Length && binPlay[i] == '1')

{

//concat += '1';

grupos.Add("1");

i++;

}

//if (concat.Length > 0)

//{

// grupos.Add(concat);

//}

}

string[] partes = grupos.ToArray();

Console.WriteLine();

foreach (string elem in partes)

{

Console.Write(elem + " ");

}

Console.WriteLine();

for (i = partes.Length - 1; i>=1; i--)

{

if (partes[i].Contains('0'))

{

string temp = partes[i];

partes[i] = partes[i - 1];

partes[i - 1] = temp;

i--;

}

}

Console.WriteLine();

foreach (string elem in partes)

{

Console.Write(elem + " ");

}

Console.WriteLine();

string binRes = "";

foreach (string elem in partes)

{

binRes += elem;

}

Console.WriteLine(binRes);

// Console.WriteLine(toBin( players) + " "+ binRes);

return BinarioADecimal(binRes);

}

static int onlySurviver1TIME\_LIMIT\_EXCEEDED(int players)

{

int res = 1;

int tope = 1;

int cont = 0;

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

{

if (i == players - 1)

{

//Console.WriteLine("res: " + res);

Console.WriteLine(toBin(i+1) + " "+ toBin(res));

return res;

//break;

}

//Console.WriteLine(toBin(i + 1) + " " + toBin(res));

res += 2;

cont++;

if (cont == tope)

{

cont = 0;

res = 1;

tope \*= 2;

}

}

Console.WriteLine(toBin(res));

return res;

}

static void Main(string[] args)

{

//string[] partes = { "1111", "00", "111" };

//for (int i = 0; i + 1< partes.Length; i+=2)

//{

// string temp = partes[i];

// partes[i] = partes[i + 1];

// partes[i + 1] = temp;

//}

//foreach (string elem in partes)

//{

// Console.Write(elem + " ");

//}

//Console.WriteLine(onlySurviver1(5));

//Console.WriteLine(onlySurviver(5));

//for (int i = 1; i < 10; i++)

//{

// Console.WriteLine(toBin(i));

//}

//Console.WriteLine(onlySurviver1(5));

Console.WriteLine( onlySurviver(13) );

//Console.WriteLine(onlySurviver1(13));

//for (int i = 1; i < 12; i++)

//{

// Console.WriteLine(toBin(i));

//}

Console.ReadLine();

}

/\*

El procedimiento es el siguiente

\* me fijo con la funcion que excede el tiempo limite las entradas y salidas

\* PERO AMBAS EN BINARIO.

\* Me doy cuenta de que los ceros se corren a la izquierda

\* Ej para players = 13, el resultado es 11

\* porque 13 en binario es 1101 y 11 en binario es 1011

\* los grupos de ceros pegados se corren a la izquierda del primer uno

\* que encuentran ej si fuera 101000 el resultado es 010001

\* por eso lo que hice fue dividir en grupos los ceros, y considerar como grupos

\* a los unos individuales, pero como otro grupo a todos los ceros pegados

\* ej para 101000 los grupos quedan: ["1","0","1","000"]

\* y queda: ["0","1","000","1"]

\* por ultimo convierto el nuevo numero binario a decimal

\*/

}

}