Fermat's factorization method is:  
If a · b = n  (where a ≤ b), then there exist some c and d such that n = c2 - d2.  
Your goal is to return for given nsuch c and d as an array.  
Since we want c and d to be uniquely determined, in all test cases n is a semiprime number.

**Example**  
For n = 15, the output should be  
fermactor(n) = [4, 1].  
15 = 42 - 12.

**Input/Output**

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

A semiprime number.

*Guaranteed constraints:*  
10 < n < 109.

* **[output] array.integer**

c and d are guaranteed to be integers if the difference between a and b is even. For all test cases, this is true.

<https://codefights.com/challenge/admES7uudjBkBNSB3/solutions>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

//FermatFactor(N): // N should be odd

// a ← ceil(sqrt(N))

// b2 ← a\*a - N

// while b2 is not a square:

// a ← a + 1 // equivalently: b2 ← b2 + 2\*a + 1

// b2 ← a\*a - N // a ← a + 1

// endwhile

// return a - sqrt(b2) // or a + sqrt(b2)

//static int[] FermatFactor(int n)

//{

// long a = (long)Math.Ceiling(Math.Sqrt(n));

// long b2 = a \* a - n;

// while (!isSquare(a))

// {

// a++;

// b2 = a \* a - n;

// }

// Console.WriteLine(a + " " + b2);

// return new int[] { (int)a,(int)b2 };

//}

//static bool isSquare(long n)

//{

// if ((long)Math.Sqrt(n) \* (long)Math.Sqrt(n) == n)

// {

// return true;

// }

// return false;

//}

static int[] fermactor(int n)

{

for (int i = 1; i <= Math.Sqrt(n) + Math.Sqrt(n); i++)

{

for (int j = i; j <= Math.Sqrt(n)+Math.Sqrt(n); j++)

{

if ((j \* j) - (i \* i) == n)

{

return new int[] { j, i };

}

}

}

return new int[] { -1 };

}

static void Main(string[] args)

{

//int n = 15;

//int n = 21;

int n = 145;

foreach (int elem in fermactor(n))

{

Console.Write(elem + " ");

}

Console.ReadLine();

}

}

}

----------otras soluciones------------

int i, j;

int[] fermactor(int n)

{

for (i = 1; ++i < n & (j = (int)Math.Sqrt(i \* i - n)) \* j != i \* i - n; ) ;

return new[] { i, j };

}

int[] fermactor(int n)

{

int c = 1, d = 0;

while (c \* c - d \* d < n)

{

c++;

d = 0;

while (c \* c - d \* d > n)

{

d++;

}

}

return new int[] { c, d };

}

int[] fermactor(int n)

{

int[] ret = new int[2];

var a = (int)Math.Sqrt(n) + 1;

var b = (int)Math.Sqrt(a \* a - n);

while ((a \* a - b \* b) != n)

{

a += 1;

b = (int)Math.Sqrt(a \* a - n);

}

ret[0] = a;

ret[1] = b;

return ret;

}