J - Buy a Shovel

Polycarp urgently needs a shovel! He comes to the shop and chooses an appropriate one. The shovel that Policarp chooses is sold for *k* burles. Assume that there is an unlimited number of such shovels in the shop.

In his pocket Polycarp has an unlimited number of "10-burle coins" and exactly one coin of *r* burles (1 ≤ *r* ≤ 9).

What is the minimum number of shovels Polycarp has to buy so that he can pay for the purchase without any change? It is obvious that he can pay for 10 shovels without any change (by paying the requied amount of 10-burle coins and not using the coin of *r* burles). But perhaps he can buy fewer shovels and pay without any change. Note that Polycarp should buy at least one shovel.

**Input**

The single line of input contains two integers *k* and *r* (1 ≤ *k* ≤ 1000, 1 ≤ *r* ≤ 9) — the price of one shovel and the denomination of the coin in Polycarp's pocket that is different from "10-burle coins".

Remember that he has an unlimited number of coins in the denomination of 10, that is, Polycarp has enough money to buy any number of shovels.

**Output**

Print the required minimum number of shovels Polycarp has to buy so that he can pay for them without any change.

**Examples**

**Input**

117 3

**Output**

9

**Input**

237 7

**Output**

1

**Input**

15 2

**Output**

2

**Note**

In the first example Polycarp can buy 9 shovels and pay 9·117 = 1053 burles. Indeed, he can pay this sum by using 10-burle coins and one 3-burle coin. He can't buy fewer shovels without any change.

In the second example it is enough for Polycarp to buy one shovel.

In the third example Polycarp should buy two shovels and pay 2·15 = 30 burles. It is obvious that he can pay this sum without any change.

#include <iostream>

using namespace *std*;

int main(void)

{

auto costOfShovel = 0, denomination = 0;

*cin* >> costOfShovel >> denomination;

auto numberOfShovels = 0;

int count = 1;

while (1)

{

numberOfShovels += costOfShovel;

if (numberOfShovels % 10 == denomination || numberOfShovels % 10 == 0)

{

*cout* << count << *endl*;

break;

}

count++;

}

return 0;

}