Find length of period in decimal value of 1/n

Given a positive integer n, find the period in decimal value of 1/n. Period in decimal value is number of digits (somewhere after decimal point) that keep repeating.

Examples:

```
Input:  n = 3
Output: 1
The value of 1/3 is 0.333333...

Input: n = 7
Output: 6
The value of 1/7 is 0.142857142857142857.....

Input: n = 210
Output: 6
The value of 1/210 is 0.00476190476190476190.....
```

Let us first discuss a simpler problem of finding individual digits in value of 1/n.

How to find individual digits in value of 1/n?
Let us take an example to understand the process. For example for n = 7. The first digit can be obtained by doing 10/7. Second digit can be obtained by 30/7 (3 is remainder in previous division). Third digit can be obtained by 20/7 (2 is remainder of previous division). So the idea is to get the first digit, then keep taking value of (remainder * 10)/n as next digit and keep updating remainder as (remainder * 10) % 10. The complete program is discussed here.

How to find the period?
The period of 1/n is equal to the period in sequence of remainders used in the above process. This can be easily proved from the fact that digits are directly derived from remainders.
One interesting fact about sequence of remainders is, all terns in period of this remainder sequence are distinct. The reason for this is simple, if a remainder repeats, then it's beginning of new period.

The following is C++ implementation of above idea.

```
// C++ program to find length of period of 1/n
#include <iostream>
#include <map>
using namespace std;

// Function to find length of period in 1/n
int getPeriod(int n)
{
   // Create a map to store mapping from remainder
   // its position
   map<int, int> mymap;
   map<int, int>::iterator it;

   // Initialize remainder and position of remainder
   int rem = 1, i = 1;

   // Keep finding remainders till a repeating remainder
   // is found
   while (true)
```

```
    {
        // Find next remainder
        rem = (10*rem) % n;

        // If remainder exists in mymap, then the difference
        // between current and previous position is length of
        // period
        it = mymap.find(rem);
        if (it != mymap.end())
            return (i - it->second);

        // If doesn't exist, then add 'i' to mymap
        mymap[rem] = i;
        i++;
    }

    // This code should never be reached
    return INT_MAX;
}

// Driver program to test above function
int main()
{
    cout <<  getPeriod(3) << endl;
    cout <<  getPeriod(7) << endl;
    return 0;
}
```

Output:

```
1
6
```

We can avoid the use of map or hash using the following fact. For a number n, there can be at most n distinct remainders. Also, the period may not begin from the first remainder as some initial remainders may be non-repetitive (not part of any period). So to make sure that a remainder from a period is picked, start from the (n+1)th remainder and keep looking for its next occurrence. The distance between (n+1)'th remainder and its next occurrence is the length of the period.

```
// C++ program to find length of period of 1/n without
// using map or hash
#include <iostream>
using namespace std;

// Function to find length of period in 1/n
int getPeriod(int n)
{
    // Find the (n+1)th remainder after decimal point
    // in value of 1/n
    int rem = 1; // Initialize remainder
    for (int i = 1; i <= n+1; i++)
          rem = (10*rem) % n;

    // Store (n+1)th remainder
    int d = rem;

    // Count the number of remainders before next
    // occurrence of (n+1)'th remainder 'd'
    int count = 0;
    do {
       rem = (10*rem) % n;
       count++;
    } while(rem != d);

    return count;
}
```

```
// Driver program to test above function
int main()
{
    cout <<  getPeriod(3) << endl;
    cout <<  getPeriod(7) << endl;
    return 0;
}
```

Output:

```
1
6
```

Reference:

Algorithms And Programming: Problems And Solutions by Alexander Shen