TOTIENT PERMUTATION

```
#include <bits/stdc++.h>
using namespace std;

bool IsPermutation(int a, int b)
{
   int digits[10] = {0};

   while(a)
   {
      unsigned int digit_a = a % 10;
      unsigned int digit_b = b % 10;

      digits[digit_a]++;
      digits[digit_b]--;

      a /= 10;
      b /= 10;
   }
   return (count(digits, digits + 10, 0) == 10);
}

int main()
{
   vector<int> totient(10000001);

   for(int i = 1; i <= 10000000; i++)
   {
      totient[i] = i;
   }
   for(int i = 2; i <= 10000000; i++)
   {
```

```
    if(totient[i] == i)
    {
        for(int j = i*2; j <= 10000000; j += i)
        {
            totient[j] = (totient[j] / i) * (i-1);
        }
    }
}
int power = 10;
double mn = 1e9;

vector<int> nums;

int N;
cin >> N;

for(int i = 1; i <= 10000000; i++)
{
    if(i == N)
    {
        cout << nums.back() << "\n";
        return 0;
    }

    if(i == power) power *= 10;
    if(totient[i] < power / 10 || totient[i] >= power) continue;
    if(i == totient[i]) continue;

    int num = i;
    int tot = totient[i];
```

```
        double ratio = (double)i / (double)tot;

        if(ratio < mn)
        {
            if(IsPermutation(num, tot))
            {
                mn = ratio;
                nums.push_back(num);

                cerr << i << ": " << tot << " (" << ratio << ")\n";
            }
        }
    }
    return 0;
}
```