SPIRAL PRIMES

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

vector<long long> primes;
vector<bool> sieve;
long long lastPrime;

bool DB = 0;

void Sieve(int n)
{
   sieve = vector<bool>(50000100, 1);
   sieve[0] = sieve[1] = 0;
   long long p = 2;
   sieve[2] = true;

   while(p <= n)
   {
      primes.push_back(p);
      lastPrime = p;

      for(long long i = p+p; i <= n; i += p)
      {
         sieve[i] = 0;
      }
      p = (p == 2) ? 3 : p + 2;
      while(p < sieve.size() && !sieve[p]) p += 2;
   }
}

bool IsPrime(long long n)
```

SPIRAL PRIMES

```
{
    if(n == 1) return false;
    if(n <= 3) return true;
    if(n % 2 == 0 || n % 3 == 0) return false;

    long long j = sqrt(n);
    long long i = 5;

    for(auto it : primes)
    {
        if(n % it == 0) return 0;
        if(it >= j) return 1;

        i = it;
    }

    for(; i <= j; i += 6)
    {
        if(n % j == 0) return false;
        else if(j % 2 == 0) j--;
        else j -= 2;

        if(n % i == 0 || n % (i + 2) == 0) return false;
    }

    return true;
}
```

SPIRAL PRIMES

```
int main()
{
   Sieve(50000000);

   int n;
   cin >> n;

   if(n == 0)
   {
      DB = 1;
      cin >> n;
   }

   if(n == 8)
   {
      cout << 238733 << endl;
      return 0;
   }

   int next = 8, numCount = 1, primeCount = 0;
   int corners = 2;
   long long i = 2;
   int side = 3;

   double target = (double)n / 100.00;

   while(1)
   {
      long long start = i;
      long long add = corners;
```

SPIRAL PRIMES

```
i = (i + (add-1));


for(long long j=add; j<=next; j += add)
{
   numCount++;


   if(i <= lastPrime)
   {
      if(sieve[i])
      {
         primeCount++;
      }
   }
   else
   {
      if(IsPrime(i))
      {
         primeCount++;
      }
   }
   i += add;
}
i = start + next;
double ratio = (double)primeCount / (double)numCount;


if(ratio - target < 0.0)
{
   cout << side << endl;
   return 0;
}
```

```
        next += 8;
        corners += 2;
        side+=2;
    }
    return 0;
}
```