Prime factorization

Time complexity is O(sqrt(n))

```
/*
   Straight-forward Prime Factorization.

   Factorizes an integer n in O(sqrt(n)).

   This can be further optimized by first precomputing all prime numbers
   up to sqrt(n) (e.g. using Sieve of Erastosthenes) and then
   checking divisibility for these primes.
   Another optimization would be to use Euler's theorem (power modulo).

  (c) 2015 Josef Ziegler

*/

#include <cstdio>
#include <cmath>
using namespace std;

void prime_factorization(int x, int* factor, int& l){
  l= 0;
  while (x%2 == 0){
    x /= 2;
    factor[l++] = 2;
  }
  int f = 3, limit = sqrt(x)+1;
  while (f < limit){
    if (x%f == 0){

        x /= f;
    factor[l++] = f;
      }
      else {
    f += 2;
```

```
    limit = sqrt(x)+1;
      }
  }
  if (x > 1) factor[l++] = x;
}

int main(){
  int A[10000];
  int l;
  for (int i=1; i<=10000; ++i){
    prime_factorization(i, A, l);
    printf(" ******** %d ******** \n",i);
    for (int j=0; j<l; ++j) printf("%d ",A[j]);
    printf("\n");
  }
}
```