```
#include <iostream>
#include <stdio.h>
#include <cmath>
#include <complex>

using namespace std ;
#define M_PI 3.14159265358979323846

typedef std::complex<double> dcomplex ;
const dcomplex di = dcomplex(0.0,1.0) ;

int prime(int atester)
{
  bool pastrouve=true;
  unsigned long k = 2;
  if (atester == 1) return 0;
  if (atester == 2) return 1;
  if (atester == 3) return 1;
  if (atester == 5) return 1;
  if (atester == 7) return 1;
  while (pastrouve)
    {
      if ((k * k) > atester) return 1;
      else
      if ((atester % k) == 0) {
        return 0 ;
      }
      else k++;
    }
}

int main (int argc, char* argv[])
{
  int np, nmax ;
  int premier[1000000] ;
  dcomplex somme ;
  int k ;

  nmax = 100 ;
  premier[1] = 2 ;
  np = 2;
  for (int k = 3 ; k <= nmax ; k=k+2) {
    if (prime(k)) {
      premier[np++] = k ;
    }
  }
  //std::cout << np << " nombres premiers :\n" ;
  //for (k = 1 ; k <= np ; ++k) {
  //  std::cout << premier[k] << "  " ;
  //}
  //std::cout << "\n\n" ;

  somme = 0.0 ;
  for (k = 1 ; k < np ; ++k) {
    somme =  somme + exp(2.0 * di * M_PI / (double) premier[k]) ;
    //if ((k % 1000) == 0)
    std::cout << premier[k] << " --> " << somme << "\n" ;
  }
}
```