// A quick and easy way to parallelize using a simple OpenMP command

C++ Program to compute \( \int k \cos(k x) \, dx \) for \( x \) on \([0, \pi/2]\)
(which simply equals \( \sin(k \pi/2) \)), but where the integrand has been "sabotaged" with \( k x = \arccos(\cos(kx)) \) in order to make the computation take much longer. We will try out some tricks to make the computation go faster -- assume that we are interested in the result of the integrand for many different \( k \)-values.

Programmer: Chris Orban  
orban@mps.ohio

Revision history:
01/30/2011  original C++ version
03/05/2011  documentation and formatting edits by Dick Furnstahl (furnstahl.1@osu.edu)
01/30/2011  original C++ version

Notes:
* This is only useful if run on a machine that can share memory with multiple processors. For example, a single computer with more than one core.
* To use g++, compile with g++ -fopenmp -o simpson_cosint_openmp simpson_cosint_openmp.cpp -lgomp
This will work fine on a MacBook, for example.
* If using Smith1094 computers on Linux you can also use the Intel C++ compiler. Load the Intel compilers with this command: module load intel-12.1.3-64
With the intel compiler loaded, compile the program like this:
icpc -openmp -o simpson_cosint_openmp simpson_cosint_openmp.cpp

#include <iostream>
#include <iomanip>
#include <fstream>
#include <cmath>

double (\*my_integrand)(double k, double x);

int main (void)
{
    double Pi = 4.*atan(1.);
    int num_i = 100001;
    int kmin = 0.0001;
    int kmax = 50.0;
    const int numi = 1000;
    int n=1;
    
    ofstream out("cosint.dat"); // open the output file
    out.open("cosint.dat"); // open the output file
    out.precision (18);
    out.setf (ios::scientific, ios::floatfield);
    exact = sin(k[i]*Pi/2.);

    return (sum);
}