Numerical Summation of a Series

(From ZOJ Problem Set – 1007)

Requirement Time Limit: 10 Seconds Memory Limit: 32768 KB

Produce a table of the values of the series

$$\psi(x) = \sum_{k=1}^{+\infty} \frac{1}{k(k+x)}$$

for the 2001 values of *x*, *x*= 0.000, 0.001, 0.002, ..., 2.000. All entries of the table must have an absolute error less than 0.5e-12 (12 digits of precision). This problem is based on a problem from Hamming (1962), when mainframes were very slow by today's microcomputer standards.

Hint

The problem with summing the sequence in equation 1 is that too many terms may be required to complete the summation in the given time. Additionally, if enough terms were to be summed, roundoff would render any typical double precision computation useless for the desired precision.

To improve the convergence of the summation process note that

$$\frac{1}{k(k+1)} = \frac{1}{k} - \frac{1}{k+1}$$

which implies y(1)=1.0. One can then produce a series for y(x) - y(1) which converges faster than the original series. This series not only converges much faster, it also reduces roundoff loss.

This process of finding a faster converging series may be repeated to produce sequences which converge more and more rapidly than the previous ones.

The following inequality is helpful in determining how may items are required in summing the series above.

$$\sum_{k=n}^{+\infty} \frac{1}{k^r} < \int_{n-1}^{+\infty} \frac{1}{x^r}\, dx \quad \text{for } k > 1 \text{ and } r \geq 1$$

```
#include <stdio.h>
#include <math.h>
#include <malloc.h>

int main()
{
	double dX = 0.000;
	double dSum = 0.0;

	double dStror1 = 0.0;
	int i, j;

	for(i=1400000; i>10000; i--)	//这里需要倒序，主要是 double 精度的问题。
	{
		dStror1 += 1/((double)(i)*(double)(i)*(double)(i+1));	//预先计算 10000 以后的和
	}

	for(i=0; i<2001; i++)
	{
		dSum = dStror1;

		for(j=10000; j>=1; j--)
		{
			dSum +=	1/(((double)(j)*(double)(j+1))*double(j+dX));	//计算 10000 以内的
		}

		dSum *= (1-dX);
		printf("%5.3f %16.12f\n", dX, 1+dSum);

		dX += 0.001;
	 }

	 return 0;
}
```