Utah High School Supercomputing Competition
2013 Software Optimization Challenge Definition
10 November 2012

Roundness:
Hoffman (1998, p. 90) calls the sum of the exponents in the prime factorization of a number its roundness. The first few values for n=1, 2, ... are 0, 1, 1, 2, 1, 2, 1, 3, 2, 2, ... (Sloane's A001222). From: http://mathworld.wolfram.com/Roundness.html

This year's UHSSC challenge requires that each team write an application to evaluate the "roundness" of a range of integers from a given minimum value to a given maximum value, where the minimum value is greater than zero and the maximum value is less than 10 million. The application should sort these numbers first by roundness, then by numerical value.

This sorted list of integers should then be written to a disk file named "uhssc.out". An example follows for the range of numbers from 7 to 15:

Number	Prime Factors	Roundness	"uhssc.out"
7	7	1	8
8	2,2.2	3	12
9	3,3	2	9
10	2,5	2	10
11	11	1	14
12	2,2,3	3	15
13	13	1	7
14	2,7	2	11
15	3,5	2	13

The output file will be compared using "`diff`" to the corresponding file produced by the reference code "UHSSCref" (included below). Four different ranges of integers will be tested, each range four times. The cumulative execution time will be recorded. The team that produces the correct output files with the least cumulative execution time will be declared the winner of the Software Optimization Challenge.

This application will be called "`UHSSCopt`". The command line for this application should look like:

```
time UHHSSCopt <min> <max>
```

where "time" is the linux time command, <min> is the smallest number in the range and <max> is the largest number in the range with 0 < min < max <= 10000000 .

```
//=========================================
//
// UHSSCref.c - UHSSC Software Optimization
//              Reference Code
//
// usage: "UHSSCopt <min> <max>"
// where <min> is the smallest
// and <max> is largest number to be factored
//
// generates a prime factorization listing starting
// at <main> and going to <max> in the following format:
//
// <integer> <pf1>,<pf2>,<pf3>,...,(<r>)
//
// where
// <pf1> = first prime factor
// <pf2> = second prime factor, etc...
// (<r>) = "roundness", the number of prime factors
//
// this listing is NOT required for the UHSSC
// Software Optimization Challenge
//
// It also produces a list of integers sorted first
// by roundness, then magnitude of the integers
// in this format:
//
// <integer> (<roundness>)
//
// this listing IS required for the UHSSC
// Software Optimization Challenge
//
//=========================================

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define MIN 2
#define MAX 1000000

int main(int argc, char **argv) {

       int number,number2,swap,factor,residual,min,max;
       int roundness[MAX];
       min=MIN;
       max=MAX;

       // get min and max of range
       if(argc>0) min=atoi(argv[1]);
       if(argc>1) max=atoi(argv[2]);
       if(min<MIN) min=MIN;
       if(max>MAX) max=MAX;
       if(min>=max) min=max-1;

       printf("\nPrime Factorization Table:");
       printf("\n<integer> <F1>,<F2>,... (<roundness>)");
       printf("\n(Not required for UHSSC)\n");

       for(number=min;number<=max;number++)
       {
              printf("\n%d  ",number);
              factor=2;
```

```
		residual=number;
		roundness[number]=0;
		while(residual>1)
		{
			if(residual%factor==0){
				//factor divides evenly
				roundness[number]++;
				if(residual!=number) printf(",");
				printf("%d",factor);
				residual/=factor;
			}
			else
			{
				// factor does not divide evenly
				factor++;
			}
		}
		printf(" (%d)",roundness[number]);
	}
	printf("\n\n");

	//
	// sort the integers first by roundness, then by magnitude
	// first, combine the integer and the roundness
	//
	for(number=min;number<=max;number++)
	{
		roundness[number]*=MAX;
		roundness[number]+=number;
	}
	//
	// sort the resulting array using a simple bubble sort
	//
	for(number=min;number<max-1;number++)
	{
		for(number2=min;number2<max+min-number;number2++)
		{
			if(roundness[number2]>roundness[number2+1])
			{
				// swap entries
				swap=roundness[number2];
				roundness[number2]=roundness[number2+1];
				roundness[number2+1]=swap;
			}
		}
	}
	//
	// print the results
	//
	printf("\nIntegers sorted by roundness:");
	printf("\n<integer> (<roundness>)");
	printf("\n(Required for UHSSC)\n");
	for(number=min;number<=max;number++)
	{
		printf("\n%d (%d)",roundness[number]%MAX,roundness[number]/MAX);
	}
	printf("\n\n");
}
```

Output resulting from the command: "`./UHSSCref 7 20`"

```
Prime Factorization Table:
<integer> <F1>,<F2>,... (<roundness>)
(Not required for UHSSC)

7  7 (1)
8  2,2,2 (3)
9  3,3 (2)
10  2,5 (2)
11  11 (1)
12  2,2,3 (3)
13  13 (1)
14  2,7 (2)
15  3,5 (2)
16  2,2,2,2 (4)
17  17 (1)
18  2,3,3 (3)
19  19 (1)
20  2,2,5 (3)


Integers sorted by roundness:
<integer> (<roundness>)
(Required for UHSSC)

7 (1)
11 (1)
13 (1)
17 (1)
19 (1)
9 (2)
10 (2)
14 (2)
15 (2)
8 (3)
12 (3)
18 (3)
20 (3)
16 (4)
```