Problem B

Mutually Friendly Numbers

Two numbers are *mutually friendly* if the ratio of the sum of all divisors of the number and the number itself is equal to the corresponding ratio of the other number. This ratio is known as the abundancy of a number. For example, 30 and 140 are friendly, since the abundancy of these two numbers is equal. Figure B.1 show this example.

$$\frac{1+2+3+5+6+10+15+30}{30} = \frac{72}{30} = \frac{12}{5}$$

$$\frac{1+2+4+5+7+10+14+20+28+35+70+140}{140} = \frac{336}{140} = \frac{12}{5}$$

Figure B.1 – 30 and 140 are friendly.

This problem consists in finding all pairs of natural numbers that are mutually friendly within the range of positive integers provided to the program at the start of the execution.

Write a parallel program to compute mutually friendly numbers.

Input

The input contains several test cases. Each line contains two integers ($1 \leq S, E < 2^{20}$) that correspond to the range where the mutually friendly numbers will be searched. The test case ends when $S=0$ and $E=0$.

The input must be read from the standard input.

Output

The output contains a message for each mutually friendly numbers found, for each test case.

The output must be written to the standard output.

Example

Input	Output for the input
30 140 100 1000 0 0	Number 30 to 140 30 and 140 are FRIENDLY Number 100 to 1000 102 and 476 are FRIENDLY 114 and 532 are FRIENDLY 120 and 672 are FRIENDLY 135 and 819 are FRIENDLY 138 and 644 are FRIENDLY 150 and 700 are FRIENDLY 174 and 812 are FRIENDLY 186 and 868 are FRIENDLY 240 and 600 are FRIENDLY 864 and 936 are FRIENDLY

```
int gcd(int u, int v) {
	if (v == 0)
		return u;
	return gcd(v, u % v);
}

void friendly_numbers(long int start, long int end) {
	long int last = end - start + 1;

	long int *the_num;
	the_num = (long int*) malloc(sizeof(long int) * last);
	long int *num;
	num = (long int*) malloc(sizeof(long int) * last);
	long int *den;
	den = (long int*) malloc(sizeof(long int) * last);

	long int i, j, factor, ii, sum, done, n;

	for (i = start; i <= end; i++) {
		ii = i - start;
		sum = 1 + i;
		the_num[ii] = i;
		done = i;
		factor = 2;
		while (factor < done) {
			if ((i % factor) == 0) {
				sum += (factor + (i / factor));
				if ((done = i / factor) == factor)
					sum -= factor;
			}
			factor++;
		}
		num[ii] = sum;
		den[ii] = i;
		n = gcd(num[ii], den[ii]);
		num[ii] /= n;
		den[ii] /= n;
	} // end for

	for (i = 0; i < last; i++) {
		for (j = i + 1; j < last; j++) {
			if ((num[i] == num[j]) && (den[i] ==
den[j]))
				printf("%ld and %ld are FRIENDLY\n",
the_num[i], the_num[j]);
		}
	}

	free(the_num);
	free(num);
	free(den);
}

int main(int argc, char **argv) {
	long int start;
	long int end;

	while (1) {
		scanf("%ld %ld", &start, &end);
		if (start == 0 && end == 0)
			break;
		printf("Number %ld to %ld\n", start, end);
		friendly_numbers(start, end);
	}

	return EXIT_SUCCESS;
}
```