Appendix B

Another old mid-exam (2016)

Problem 1: Reversible multiples of seven

A positive integer that is divisible by 7, while the reversal of its digits is also divisible by 7 is called a *reversible multiple of seven*. For instance, 259 is such a number because $259 = 7 \times 37$ and $952 = 7 \times 136$. Note that leading zeros in the reversal are ignored, so the reversal of 700 is 7. Hence, 700 is also a reversible multiple of seven.

The input of this problem consists of two integers `a`, and `b` such that $1 \leq$ `a` $\leq$ `b` $\leq 10000000 = 10^7$. The output should be the number of integers `x` such that `a` $\leq$ `x` $\leq$ `b` and `x` is a reversible multiple of seven. Note that there are no spaces in the output, and that the output ends with a newline (`\n`):

Example 1:
input:
```
1 1000
```
output:
```
21
```

Example 2:
input:
```
777 800
```
output:
```
1
```

Example 3:
input:
```
700 705
```
output:
```
1
```

Problem 2: Pangram

A *Pangram* is a sentence using every letter of the alphabet at least once. A famous English pangram is "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG".

Write a program that reads from the input a sentence, and outputs whether the sence is a *pangram*, or *no pangram*. The input consists of a sentence containing spaces and uppercase letters from the conventional 26 letter alphabet ('A', 'B', .., 'Z'). The sentence ends with a dot ('.'). Make sure that the output of your program has the same format as in the following examples. Note that there are no spaces in the output, and that the output ends with a newline (`\n`):

Example 1:
input:
```
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.
```
output:
```
PANGRAM
```

Example 2:
input:
```
AMAZINGLY FEW DISCOTHEQUES PROVIDE JUKEBOXES.
```
output:
```
PANGRAM
```

Example 3:
input:
```
THIS SENTENCE IS CLEARLY NOT A PANGRAM.
```
output:
```
NO PANGRAM
```

Problem 3: Hamming numbers

Hamming numbers are integers n of the form $n = 2^i \times 3^j \times 5^k$.
For example, 360 is such a number because it can be divided three times by 2, two times by 3 , and once by 5 (i.e. $360 = 2 \times 2 \times 2 \times 3 \times 3 \times 5$). The number 2520 is not a Hamming number, because it has a factor 7 ($2520 = 360 \times 7$).

The first ten Hamming numbers are:

$$\begin{aligned} 1 &= 2^0 \times 3^0 \times 5^0 \\ 2 &= 2^1 \times 3^0 \times 5^0 \\ 3 &= 2^0 \times 3^1 \times 5^0 \\ 4 &= 2^2 \times 3^0 \times 5^0 \\ 5 &= 2^0 \times 3^0 \times 5^1 \end{aligned} \qquad \begin{aligned} 6 &= 2^1 \times 3^1 \times 5^0 \\ 8 &= 2^3 \times 3^0 \times 5^0 \\ 9 &= 2^0 \times 3^2 \times 5^0 \\ 10 &= 2^1 \times 3^0 \times 5^1 \\ 12 &= 2^2 \times 3^1 \times 5^0 \end{aligned}$$

Of these numbers, the numbers 4, 6, 9, and 10 have a sum of the exponents that equals 2 ($i+j+k = 2$).

The input of this problem consists of three integers `a`, `b`, and `n` such that $1 \leq$ `a` $\leq$ `b` $\leq 10000000 = 10^7$. The output should be the number of Hamming numbers `x` (where `a` $\leq$ `x` $\leq$ `b`) of which the sum of the exponents equals n. Note that there are no spaces in the output, and that the output line must end with a newline (`\n`):

Example 1:
input:
`1 12 2`
output:
`4`

Example 2:
input:
`1 12 1`
output:
`3`

Example 3:
input:
`1 1000000 10`
output:
`57`

Problem 4: Smith numbers

A *prime number* is a natural number greater than 1 that has no positive divisors other than 1 and itself. A *composite number* is a natural number that is not a prime number. Each composite number can be uniquely expressed as a product of prime numbers. This product is called its *prime factorization*.

A *Smith* number is a *composite number (i.e. a non-prime number)* greater than 1 for which the sum of its digits is equal to the sum of the digits in its prime factorization.

For example, 825 is a Smith number, because its prime factorization is $825 = 3 \times 5 \times 5 \times 11$, and $8 + 2 + 5 = 3 + 5 + 5 + 1 + 1$.

Write a program that reads from the input a positive integer, and outputs `YES` if the number is a Smith number, and `NO` otherwise. Note that there are no spaces in the output, and that the output ends with a newline (`\n`):

Example 1:
input:
`825`
output:
`YES`

Example 2:
input:
`42`
output:
`NO`

Example 3:
input:
`4937775`
output:
`YES`

Problem 5: Takuzu checker

A *Takuzu* is a number placement puzzle. The objective is to fill an 8×8 grid with 1s and 0s, where there is an equal number of 1s and 0s in each row and column (hence four 0s and four 1s) and no more than two of either number adjacent to each other. Moreover, there can be no identical rows, nor can there be identical columns. An example of a Takuzu puzzle and its solutions are given in the following figure.

0				**0**		**0**	
0					**1**		
			1			**0**	
		0		**0**			
							1
0		**0**		**0**	**0**		
						1	
	1						

0	0	1	1	**0**	1	**0**	1
0	1	1	0	0	**1**	1	0
1	0	0	**1**	1	0	**0**	1
1	1	**0**	1	**0**	0	1	0
0	0	1	0	1	1	0	**1**
0	1	**0**	1	**0**	**0**	1	1
1	0	1	0	1	0	**1**	0
1	**1**	0	0	1	1	0	0

Write a program that reads from the input a completely filled 8×8 grid of 0s and 1s. There are 8 input lines, one for each row. A row consists of 8 characters ('`0`' and '`1`'), followed by a newline ('`\n`'). Your program should output `CORRECT` if the grid satisfies all the rules of a Takuzu puzzle, otherwise it should output `INCORRECT`. Of course, your output must be a a single line, without spaces, than ends with a newline (`\n`).

Example 1:
input:
```
00110101
01100110
10011001
11010010
00101101
01010011
10101010
11001100
```
output:
```
CORRECT
```

Example 2:
input:
```
00110111
01100110
10011001
11010010
00101101
01010011
10101010
11001100
```
output:
```
INCORRECT
```

Example 3:
input:
```
00110101
01100110
10011001
11010010
00110101
01010011
10101010
11001100
```
output:
```
INCORRECT
```