Problem 0. Race Track

A **racetrack** is formed by two rectangles, an outer rectangle and an inner rectangle (the **field**). The outer rectangle has a length L and a width W. The inner rectangle has a length i with $i < L$ and a width m with $m < W$. The inner width m can be computed from L, i, and W because the breadth of the track is the same on all four sides. All lengths are given in feet. Building the surface of a race track costs \$8.00 per square-foot. The inner rectangle is **not** surfaced.

Write a program with the following input and output:

The **input** file will have the following format: The first line is an integer which tells you how many subsequent lines there are in the file. Each subsequent line in the file contains three positive integers, L, W, and i in this order, separated by one or several blanks. L, W, and i are positive integers < 300.

Your **output** file should have the following content: For each line (after the first) in the input file, there should be a corresponding line in the output file giving L, W, and i and the total cost in Dollars to build the track.

For example, if the input file contains these three lines:

```
2
200 180 160
220 180 170
```

then, the following would be the correct output file:

```
200 180 160 108800
220 180 170 140000
```

Problem 1. Being superb

A positive integer n is **superb** if and only if the sum of all its divisors (excluding n itself) is greater than n. For example, 12, 18, and 100 are superb because

$$\begin{aligned} 1+2+3+4+6 &> 12\ , \\ 1+2+3+6+9 &> 18\ , \\ 1+2+4+5+10+20+25+50 &> 100\ . \end{aligned}$$

But, for example, 4 is not superb because its divisors are 1 and 2 and their sum is less than 4; 6 is not superb because its divisors, 1, 2 and 3, sum to 6.

Write a program with the following input and output:

The **input** file will have the following format: The first line is an integer which tells you how many subsequent lines there are in the file. Each subsequent line in the file contains a positive integer n in the range $2 \leq n \leq 1000000$ (1 million).

Your **output** file should have the following content: For each n in the input file, there should be a corresponding line in the output file giving n and either the word `superb` or the word `non-superb`. (The number n and the word `superb`/`non-superb` should be separated by a space.)

For example, if the input file contains these four lines:

```
4
24
4
100
1000000
```

then, the following would be the correct output file:

```
24 superb
4 non-superb
100 superb
1000000 superb
```

Problem 2. Calendar

An annual calendar has an **annual starting day**, which is the weekday on which January 1st falls. A year may be a leap year (February with 29 days) or a normal year (February with 28 days). An **appointment date** is given by a month (coded as a number) and a day in the month (a number between 1 and 30, or between 1 and 31, except for February). The problem is to find the weekday on which an appointment date falls.

Write a program that takes as input whether the year is a leap year, the annual starting day and an appointment date and computes on which weekday the appointment date falls.

Weekdays are coded as follows:

```
Sunday = 1
Monday = 2
Tuesday = 3
Wednesday = 4
Thursday = 5
Friday = 6
Saturday = 7
```

Your **input** file has the following format: The first line contains a positive integer $C \leq 20$, which is the number of lines following in this file. Each subsequent line has the following format. The first number is "1" for a leap year, and "0" for a normal year. The second number stands for the weekday that is the annual starting day. The third number is the month of the appointment date. The fourth number is the day within the month of the appointment date.

Your **output** file should contain a copy of the input line, followed by the weekday as a word.

For example, if the input file has as contents

```
2
0 7 1 14
0 4 11 25
```

then, the contents of the output file should look as follows:

```
0 7 1 14 Friday
0 4 11 25 Tuesday
```

PROBLEM 3. SET INTERSECTION

Given are 3 sets of positive integer numbers. For simplicity we assume that each set fits into one single 80 character line of the input file. Numbers are separated by one or more blanks. As a further simplification, all numbers are between 0 and 100. Numbers are in no particular order. Numbers may occur repeatedly.

The set intersection of three sets of numbers contains every number that occurs in all three sets. If the number occurs repeatedly in all three sets, it still may occur only **once** in the intersection. The result (intersection) should appear in ascending sorted order.

The **input** file has the following format: The first line of the file is an integer indicating how many subsequent lines are in the file. This number may be assumed to be divisible by three. The next three lines are the three sets of integers. Every subsequent group of three lines defines one intersection problem.

Consider, for example, an input file that contains these lines:

```
6
45 23 16 23 89 45
23 12 89 100 55 23 45
66 77 23 23 45
66 77 88 99 66
77 66 99 88 66
77 99 66 66 77
```

Your **output** file should contain one line for the intersection of each consecutive group of three sets.
In our example, the output will be:

```
23 45
66 77 99
```

PROBLEM 4. COLOR-BLIND POKER

Color-blind poker is a simplified version of poker with a standard 4-suit, 52-card deck, in which the suits are ignored. The ranking of the cards, from highest to lowest, is: `A K Q J 10 9 8 7 6 5 4 3 2`. Each player has a hand with 5 cards. The players compare their hands according to the following rankings, listed from best to worst:

1. **Four of a kind:** four cards of the same rank accompanied by a "kicker," for example `44442`. If two hands have four of a kind, the higher four wins, so that `44442` beats `3333K`.
2. **Full house:** three cards of one rank accompanied by two of another, such as `777JJ`. If two hands have a full house the higher triplet wins, so that `44422` beats `333AA`.
3. **Straight:** five cards in sequence, such as `76543`. If two hands have a straight, the higher ranked top card wins (note that the ace is always high and that "around the corner" straights are not allowed, so that neither `5432A` nor `32AKQ` is a straight).
4. **Three of a kind:** three cards of the same rank and two kickers, such as `KKK84`.
5. **Two pairs:** two cards of one rank, two cards of another rank and a kicker of a third rank, such as `KK449`, ranked by the top pair, then the bottom pair and finally the kicker, so that `KK449` beats any of `QQJJA`, `KK22Q`, and `KK445`.
6. **One pair:** two cards of one rank accompanied by three kickers of different ranks, such as `AAK53`; ranked by the pair, followed by each kicker in turn, so that `AAK53` beats `AAK52`.
7. **Empty hand:** the remaining hands — not ranked.

Write a program that compares two hands and decides which one wins. The **input** file will have the following format: The first line is an integer which tells you how many subsequent lines there are in the file. Each subsequent line describes two hands in a format of five `char`s, a space, and then another five `char`s (no suit is given). Every `char` is either a digit in $\{0, 2, 3, 4, 5, 6, 7, 8, 9\}$ or one of the letters `A`, `K`, `Q`, `J`; digit 0 corresponds to card `10`; all other `char`s correspond to the cards in the natural way. Your **output** file should have the following content: For each two hands in the input file, there should be a corresponding line in the output file giving the hands (in identical format as there were in the input) and then, after a space, one of three numbers 0, 1, or 2, depending on whether both hands have identical ranks (tie), or hand 1 is winning, or hand 2 is winning, respectively. For example, if the input file contains these lines:

```
            5
            02020 JJQQA
            02345 00234
            06978 QQQK2
            0JQ98 7890J
            23890 02345
then the following would be the correct output file:
            02020 JJQQA 1
            02345 00234 2
            06978 QQQK2 1
            0JQ98 7890J 1
            23890 02345 0
```

This is the end of Problem 4.

PROBLEM 5. TWINS

Two numbers are called **twins** if one can be obtained from another by changing the order of the digits. For example, numbers 52 and 25 are twins, numbers 123 and 231 are twins, but 123 and 234 are not, and neither are numbers 101 and 11.

Write a program with the following input and output:

The **input** file will have the following format: The first line is an integer N which tells you how many subsequent lines there are in the file. You can assume that $1 \leq N \leq 100$. Each subsequent line in the file contains a positive integer M in the range $1 \leq M \leq 999$.

Your **output** file should contain all numbers M from the input that are not twins to any other number from the input; each number should be put on a separate line and the numbers should be put in the order they appear in the input.

For example, if the input file contains these lines:

```
11
218
4
101
77
82
281
333
182
11
110
28
```

then, the following would be the correct output file:

```
4
77
333
11
```

PROBLEM 6. WINNING TICKET

On planet Heavenland, every year on January 1, the King conducts an elimination game for the members of parliament with the winner getting 1 billion heavens. The King puts all parliament members in a row, and starting with the first person in the row, eliminates cyclically every third remaining person until only one is left — that parliament member wins the prize.

For example, if there are 7 parliament members, then in the first round member 1 will be eliminated, then member 4, then member 7, then member 5, then member 3, and finally member 6; so the winner is the parliament member who was 2nd in the row. If there are 8 parliament members, then the elimination goes: 1, 4, 7, 3, 8, 6, 2, and the winner is number 5.

Write a program that for a given number of participants will compute the winning number.

The **input** file will have the following format: The first line is an integer which tells you how many subsequent lines there are in the file. Each subsequent line in the file contains a positive integer n in the range $2 \leq n \leq 1000000$ (1 million).

Your **output** file should have the following content: For each n in the input file, there should be a corresponding line in the output file giving n and the number of the winner.

For example, if the input file contains these four lines:

```
5
2
3
7
8
12
```

then, the following would be the correct output file:

```
2 2
3 3
7 2
8 5
12 8
```

PROBLEM 7. ROBOTS

Every robot on planet Robotia has a *year* of manufacture and a *version* number, which is the year in which that version was first manufactured. A new version of robot is introduced each year, and old versions are never allowed to go out of production.

The number of robots of each version produced each year is determined as follows:

- Each year, beginning in year 0, exactly one robot of version 0 is manufactured.
- In year N, the number of robots of version M, $1 \leq M \leq N$, that are manufactured is equal to the number of robots of version M manufactured the previous year plus the number of robots of version $M-1$ manufactured the previous year.
- In each year N, there are no robots of version M for $M > N$.

Write a program that for given numbers N and M, computes the number of robots of version M manufactured in year N.

The **input** file will have the following format: The first line is an integer which tells you how many subsequent lines there are in the file. Each subsequent line in the file contains two nonnegative integers N and M (separated by a space) in the range $0 \leq M, N \leq 20$.

Your **output** file should have the following content: For each pair N and M in the input file, there should be a corresponding line in the output file giving N, M and the number of robots of version M in year N.

For example, if the input file contains the following six lines:

```
5
7 0
17 3
6 1
9 2
1 6
```

then, the following would be the correct output file:

```
7 0 1
17 3 680
6 1 6
9 2 36
1 6 0
```

Problem 8. Arbitrary Precision Integer Arithmetic

Given are two positive integer numbers that may have up to (and including) 80 digits. Compute the sum, the difference, the product and the integer quotient (with no remaninder) of these two numbers. Neither the first number nor the second number will ever be equal to zero. The result of both addition and multiplication may overflow 80 digits. In that case, your output should consist of additional lines as needed. The continuation should be marked by an “&” sign at position 80 in the first output line of a two-line output. In other words, no output line should contain more than 80 characters. The **input** file has the following format: The first line of the file is an integer indicating how many subsequent lines are in the file. This number may be assumed to be divisible by two. Each subsequent pair of lines contains a pair of integers. Consider, for example, an input file that contains these lines:

```
4
1000000000000000000000000000000000000000000000
5
10000000000000000000000000000000000000000000000000000000000000000000000000000000
100
```

Your **output** file should contain six or more lines for each group of two input numbers. The first two lines are echos of the input numbers. The third line is the sum, the fourth line the difference, the fifth line the product and sixth line the quotient. Both sum and product may overflow to another line. The second number may be larger than the first number, resulting in a negative difference and a zero quotient. In our example, the output will be:

```
1000000000000000000000000000000000000000000000
5
1000000000000000000000000000000000000000000005
999999999999999999999999999999999999999999995
5000000000000000000000000000000000000000000000
200000000000000000000000000000000000000000000
10000000000000000000000000000000000000000000000000000000000000000000000000000000
100
10000000000000000000000000000000000000000000000000000000000000000000000000000100
9999999999999999999999999999999999999999999999999999999999999999999999999999900
1000000000000000000000000000000000000000000000000000000000000000000000000000000&
000
100000000000000000000000000000000000000000000000000000000000000000000000000000
```