

## Problem 1: Code 777

In the logic game Code 777 by Robert Abbott, there are 28 cards each of which has single colored digit on its face. The deck contains the following cards: one green 1, two yellow 2s, three black 3s, four brown 4s, four red and one black 5, three pink and three green 6s, and one pink, two yellow, and four blue 7s.

There are four players: North, East, West, and South. In front of each player is a rack containing three cards; the remaining 16 cards are not used. Each player sees the nine cards on the other racks, but not the three cards on the player's own rack.

Each player must determine what cards on their rack based on the cards they can see, and on the answers that the other players give to some of the questions below.

1. On how many racks is the sum of the numbers 18 or more?
2. On how many racks is the sum of the numbers 12 or less?
3. On how many racks is there the same number in different colors, e.g., a blue 7 and a yellow 7?
4. On how many racks are there three different colors?
5. On how many racks are the numbers either all even or all uneven?
6. On how many racks are there at least two identical cards (same number and same color)?
7. On how many racks do you see three consecutive numbers?
8. How many colors do you see?
9. How many colors appear at least three times?
10. How many numbers are missing?
11. How many of the following cards do you see: green 1, black 5, pink 7?
12. Do you see more 3s or more pink 6s?
13. Do you see more green 6s or more yellow 7s?
14. Do you see more 2s or more yellow 7s?
15. Do you see more pink 6s or more green 6s?
16. Do you see more blue 7s or more 7s of other colors?
17. Do you see more brown or more blue numbers?
18. Do you see more red or more pink numbers?
19. Do you see more green or more blue numbers?
20. Do you see more yellow or more pink numbers?
21. Do you see more black or more brown numbers?
22. Do you see more black or more red numbers?
23. Do you see more green or more yellow numbers?

Note that during the game not all of the questions listed above will be asked. You may assume that the players answer the questions truthfully.

You are to write a program to play North. The first 3 lines of input to your program will describe the racks of the other players of the game. These lines will contain words and integer values separated by one or more white space characters. The first word on each line will specify the player (i.e., East, West, or South). The rest of the line will contain 3 pairs, each consisting of a word (i.e., black, blue, brown, green, pink, red, or yellow) and a number. These pairs specify the cards held by that player.

The next line of input will contain a single integer value,  $N$ , that specifies the number of questions that the program has been provided answers for. Each of the remaining  $N$  input lines will contain the name of the player who provided the answer, the number of the question that was answered, and the player's answer. Questions 1 through 11 are answered with a number, questions 12 through 23 are answered with one of three characters: < denoting fewer or the first, = denoting the same number (even none) of either, and > denoting more of the first. The answer provided by a player is based on the three racks that the answering player can see. One or more white space characters will separate items on the input lines.

The output produced by your program must describe North's rack, in the style of the first three input lines.

Example input:

```
South green 1 black 3 pink 7
East brown 4 red 5 black 5
West red 5 green 6 yellow 7
4
South 1 1
East 9 1
West 10 0
South 16 =
```

Example output:

```
North yellow 2 green 6 blue 7
```

## Problem Number 2: Spock What Time is it?

When the Federation was founded, it was decided that all members of the Federation would use a standard calendar. The Federation declared midnight on 2162-01-04 to be stardate zero. Thus stardates began on the fourth day of the year 2162, and increased at the rate of five units per day.

The Federation decided to limit the length of a stardate to four digits. As a result, what would have been stardate 10000 (midnight on 2167-06-27) was made stardate 0000 again. The first group of stardates could be referred to, when necessary, as zeroth-issue stardates, such as [0]1234, and the new issue as first-issue stardates, such as [1]1234. This reset to zero continued to occur every five and a half years, until 2266, when the 19<sup>th</sup> issue of stardates started. That year, Starfleet put together a committee to investigate what type of stardate system would be more acceptable. The committee's report, in 2267, recommended that the stardate rate be slowed to 0.1 units per day. This would make the same number of digits as had been previously used, and had covered five and a half years, cover two and a half centuries. It was decided that this system should be field-tested between stardates [19]7340 and [19]7840, i.e., 500 units, 5000 days. So from 2270-01-26 to 2283-10-05 this system was used.

This new system proved to be unpopular, and as a result, it was decided in 2280 that at the end of the test period (SD [19]7840) the new rate should not continue. Instead, a 0.5 units per day rate would be used, which would solve the main problems of both earlier systems. This system was used from stardate [19]7840, and was intended to be a permanent change. In 2318, over 150 years after the incorporation of the Federation, it was decided that starships should start to use a quad-cent calendar, which would keep the years the right length but make the day slightly longer. The quad-cent calendar eliminates leap years by assuming a standard year length of 365.2425 days. In keeping with this longer-term view of time, the stardates were increased to five digits, and the rate was changed to 1000 units per standard year. So on what would have been stardate [20]5006.0, i.e., midnight on 2323-01-01, stardates were reset to [21]00000.

The historical division of the Federation is having a difficult time converting dates to stardates. You have been asked to write a program that accepts a single date in the form *yyyy-mm-dd* (where *yyyy*=year, *mm*=month, and *dd*=day) and prints out the corresponding stardate without the issue number.

Sample input:

2372-01-11

Sample output:

49027.3

## Problem Number 3: Protect Your Space

You are defending the first octant in 3-dimensional space. You only need to consider points with all non-negative integer coordinates that do not exceed a single integer  $N$ .

To orient themselves, commanders of every octant hold out their left hand, point their thumb up, index finger straight ahead, and middle finger to the right. They align the x-axis with their middle finger, the y-axis with their index finger, and the z-axis with their thumb. The y-axis is called 'forward', the z-axis is called 'up'.

At some point in time, a single enemy spaceship will arrive in your octant at a point  $X, Y, Z$ , on one of the six faces of your bounded octant, never on an edge. Upon arrival, the ship will be oriented perpendicular to the face, looking into the bounded octant.

The spaceship has an engine in the back which propels it forward, and four thrusters labeled A, B, C, and D at the front. When the ship arrives at your octant, thruster A points 'up', or 'forward' if 'up' is impossible, and the remaining thrusters are arranged clockwise from A to D, looking from the back to the front of the ship.

The flight plan of the spaceship is based on the following actions, each of which takes one time unit:

Letter	Description
F	Fire the engine, moving to the adjacent point in the current direction.
A	Fire thruster A to change the current direction by 90 degrees, i.e., from the initial position the spaceship would then be pointed 'down' or 'backward'.
B	Fire thruster B to change the current direction by 90 degrees
C	Fire thruster C to change the current direction by 90 degrees
D	Fire thruster D to change the current direction by 90 degrees

Each of the eight corners of your bounded octant contains a missile launcher. From one of these you release a single missile when the spaceship arrives.

In each time unit, once the spaceship arrives, the following happens:

- The missile moves from its current position to that adjacent point that is closest, in terms of Euclidean distance, to the current position of the spaceship. Only one coordinate will change because the missile cannot move diagonally through space. If there is a choice, the missile will always move to the best possible point.

- Immediately after the missile starts to move, the spaceship carries out the next action from its flight plan; action F will change the position of the spaceship, the other actions will not.

If the missile and the spaceship occupy the same position at the end of a time unit, they disintegrate. Otherwise, if the spaceship is on one of the faces of your bounded octant it has escaped.

You are to write a program that, given the size of an octant and the flight plan of an enemy spaceship, determines if the ship escapes or determines the trajectory of a missile that collides with the ship and destroys it.

The input to your program will consist of three lines of input. The first line will contain the integer value  $N$ . The second line will contain three integer values, separated by one or more white space characters, that give the  $X$ ,  $Y$ , and  $Z$  coordinates of the location where the enemy ship enters the octant. The third line of input will contain a single string that specifies the flight path of the enemy ship.

Your program will print “escaped” if the enemy ship is able to leave the octant without being destroyed, or the trajectory of a missile that collides with the spaceship. The trajectory consists of one line with coordinates ( $X$ , followed by  $Y$ , followed by  $Z$ ) for each point that the missile has visited, starting with a corner of your bounded octant and ending with the point of collision.

Sample Input:

```
4
1 0 1
A
```

Sample Output:

```
escaped
```

Sample Input:

```
4
1 0 1
FDF
```

Sample Output:

```
3 0 0
2 0 0
2 1 0
2 1 1
```

## Problem Number 4: Keep It Simple Stupid

A prefix expression consists of either an unsigned integer, the variable "x", or one of the operators [+ , - , \* , /] followed by two prefix expressions. Write a program that when given a prefix expression returns an equivalent expression with the following simplifications:

- Anything times 0 is 0
- 0 divided by anything other than 0 is 0
- Anything divided by 0 is an error
- Anything plus 0 is the other thing
- Anything multiplied or divided by 1 is the other thing
- Any operation on two integer value is the value that results from performing the operation (i.e., + 2 2 simplifies to 4)

The only simplifications that you are to perform are the ones listed above. Also for the purposes of this problem you are to assume that \* and + are not commutative or associative. You may assume that one or more white space characters separate the tokens that make up the expression and that expression is contained on a single line. If the simplification results in an error (i.e. something is divided by 0) your program should print the message "Error".

Sample input:

\* x 1

Sample output:

x

Sample input:

\* 1 \* 2 3

Sample output:

6

Sample input:

/ \* x 1 - 3 2

Sample output:

x

Sample input:

/ 0 0

Sample output:

Error

## Problem Number 5: Who is Still Alive?

At various points in history a number of presidents (former and the current president) have been alive. For example, currently there are six living presidents: Ford, Carter, Reagan, George the Second (George H. Bush), Clinton, and George the Third (George W. Bush)<sup>1</sup>.

Write a program that takes as input the name, inauguration date, and date of death (if appropriate) of several presidents and produces as output the periods of time when the maximum number of current or former presidents are alive, together with the names of those presidents in alphabetical order. Note, there may be several such intervals and all must be listed in chronological order.

The input to your program will start with a line that contains a single integer,  $N$ , that specifies the number of presidents to be considered. Each of the following  $N$  lines of input describes a single president. An input line for a president will consist of an identifier consisting of letters only. The identifier will be followed by the inauguration date, and by an optional date of death. Dates will be expressed in the form  $yyyy-mm-dd$  where  $yyyy$  represents a year,  $mm$  represents a month, and  $dd$  represents a day. If the president is still alive, no death date will be given. One or more white space characters will separate tokens on the input line. Note that if a president was inaugurated more than once, they may appear in the input more than once; remember, however, that once a president, always a president.

The output from the program will specify the period of time when the most presidents were alive, followed by the names in alphabetical order. If there are multiple intervals, the intervals will be listed in chronological order, separated by a blank line.

### Sample input:

3

Nixon 1969-01-20 1994-04-22

Ford 1974-01-20

GWBush 2001-01-20

### Sample output:

1969-1-20 to 1994-4-22

Ford

Nixon

1974-1-20 to 2003-11-7

Ford

GWBush

[1] George Washington was George the First.

## Problem 6: Circular Graphs

We say that an undirected graph  $G$  on  $n$  vertices  $\{0, \dots, n-1\}$  is circular if the edge set of  $G$  can be defined by a set of distances  $DIST$  so that  $\{x, y\}$  is an edge if and only if  $|x-y| \bmod n \in DIST$ . Note that if  $d \in DIST$  then also  $(n-d) \in DIST$ . For example, a pentagon graph can be seen as a circular graph on 5 vertices with  $DIST = \{1, 4\}$ , and a 5-star has  $DIST = \{2, 3\}$ .

Let  $p$  be a prime equal to 1 modulo 4. The Paley graph  $Q_p$  is defined as a circular graph on the vertex set  $Z_p = \{0, \dots, p-1\}$  with

$$DIST = \{ d \mid \text{there exists } i \in Z_p, \text{ such that } i \neq 0 \text{ and } d = i^2 \bmod p \}$$

For example, the Paley graph  $Q_{13}$  has  $DIST = \{1, 3, 4, 9, 10, 12\}$ , and the maximum complete subgraphs in  $Q_{13}$  have 3 vertices. Equivalently,  $Q_{13}$  has triangles but no complete subgraphs on 4 vertices. A maximum complete subgraph is formed by a subset of vertices, such that every vertex is connected to every other.

Write a program that reads a single line of input that contains an integer  $p$ , where  $p$  is an odd prime less than 1000 of the form  $p = 4k+1$  and outputs the number of vertices in the maximum complete subgraph in the Paley graph  $Q_p$ .

Your program doesn't need to check if the input  $p$  is an odd prime of the form  $4k+1$ . You can also assume that your program will work only with graphs for which the maximum number of vertices in any complete subgraph is no larger than 15.

Sample input:

5

Sample output:

2

Sample input:

229

Sample output:

9

## Problem 7: Working on the Railroad

A typical railroad switchyard consists of a number of parallel tracks filled with all sorts of cars. One of the basic operations of the switchyard is to take specific cars from the parallel tracks and assemble them into a complete train. In this program you are to write a program that simulates the actions of a switchyard operator.

The first line of input processed by your program will contain a single integer,  $N$ , that specifies the number of tracks in the switchyard. The next  $N$  input lines will contain at least one identifier, where an identifier consists of a sequence of one or more letters and/or digits. If a line contains more than one identifier, one or more white space characters will separate each identifier. The first identifier on the line specifies the name of the track, and the remaining identifiers (if any) specify the names of the cars on the track. The order of the cars in the input specifies the order of the cars on the corresponding track; the second identifier on the line is the leftmost car on the track, and the last identifier is the rightmost car on the track. You may assume that the track names are unique.

Lines following the switchyard description specify an arbitrary number of commands. Each command takes one input line. One or more white space characters separate each item in the command. In the table below, *number-of-cars* is a signed integer value, and *track-name* is an identifier as defined above. The switchyard operator understands the following commands:

Command	Description
<i>track-name</i> <b>left</b> <i>number-of-cars</i>	If <i>number-of-cars</i> is 1, one car is taken off the left end of <i>track-name</i> and added to the left end of the train being assembled.  If <i>number-of-cars</i> is $-1$ , one car is taken off the left end of the train being assembled and returned to the left end of <i>track-name</i> .  For numbers other than 1 or $-1$ , the operation (i.e., left or right) is executed the specified number of times. In other words the cars are moved individually <i>not</i> as one unit.
<i>track-name</i> <b>right</b> <i>number-of-cars</i>	The same as the <b>left</b> command, except that it applies to the right end of a track and the train.
<b>end</b>	End of input.

Your program will print the sequence of cars assembled into the train once all input has been processed. Note that processing stops as soon as the **end** command is entered *or* an impossible command is detected.

If the program successfully processes all commands, it will print out the number of cars on the final train and the contents of the final train. If an attempt is made to execute an invalid command, the program will display the message “Illegal command” and terminate.

Sample input:

```
1
t0 a b c
t0 left 1
t0 right 2
end
```

Sample output:

```
The train has 3 cars
a c b
```

Sample input:

```
1
t0 a b c
t0 right 4
end
```

Sample output:

```
Illegal command
```

## Problem 8: The Game of Jawbreaker

Jawbreaker is played on an  $N$  by  $N$  matrix filled with single letters. The objective of the game is to clear as many letters off the board as possible.

A move consists of selecting a letter position that has at least one equal letter as a horizontal or vertical neighbor. The selected letter and all equal letters that touch vertically or horizontally (but not diagonally) are removed from the board. Letters slide down in each column to take up the empty positions. If a column is empty, the columns to the left slide over to the right.

You are to write a program that will play Jawbreaker. The first line of input presented to your program will contain the integer  $N$ . The next  $N$  lines of input will each contain exactly  $N$  characters and will specify the characters in the matrix.

The program will determine a sequence of moves required to remove the most characters from the matrix. The output from the program will be the minimal number of characters remaining after playing the game.

Sample input:

4

AAAA

BBBB

BBCB

BBBB

Sample output:

1