2018

Problem Packet

Problem	Point Value
Problem 1: Pass/Fail	5
Problem 2: AEIOU	5
Problem 3: SufFIX	10
Problem 4: Rock, Paper, Scissors	10
Problem 5: Collatz Conjecture	15
Problem 6: Space Station Repair Lights	15
Problem 7: Palindrome Series	15
Problem 8: Apollo 13	20
Problem 9: Are Eucliding Me?	20
Problem 10: Bishop's Move	20
Problem 11: Chroma Key Effect	25
Problem 12: Sieve of Eratosthenes	30
Problem 13: Peoplebook	30
Problem 14: ZIPPER text	30
Problem 15: Piecewise Encrypter	35
Problem 16: Twinkle Twinkle	40
Problem 17: Tic Tac Toe Checker	45
Problem 18: Dominating Disney	60
Problem 19: It's an Enigma!	70
Total Possible Points	**500**

Problem 1: Pass/Fail
5 points

CODE QUEST 2018

Input File: `Prob01.in.txt`

Introduction

Students stress about their grades. It's only natural, because grades are important! However, many universities offer the option to take certain classes as pass/fail. Generally speaking, if you get a grade of 70 or above, you pass. Otherwise, you fail.

Your job is to write a program to read in a list of grades and determine if the student earning that grade passed or failed the class.

Program Input

The first line of the file `Prob01.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single integer grade, where 0 <= grade <= 100

Example Input:

```
6
0
48
69
70
87
100
```

Program Output

For each grade, your program should print `PASS` is the grade is 70 or above, and `FAIL` if it is not.

Example Output:

```
FAIL
FAIL
FAIL
PASS
PASS
PASS
```

Problem 2: AEIOU
5 points

CODE QUEST 2018

Input File: `Prob02.in.txt`

Introduction

The English alphabet contains five vowels – A, E, I, O, and U. The other 21 letters of the alphabet are called consonants. Vowels are like the glue to the English language – without them, our words would be unpronounceable! As a result, they are together the most common letters in the English language.

Today, we need you to count how many vowels appear in a series of sentences.

Program Input

The first line of the file `Prob02.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line with a series of lowercase words separated by spaces.

Example Input:

```
3
code quest is fun
good luck programming today
queueing has five vowels in a row
```

Program Output

For each test case, your program should print out the number of vowels contained in the input.

Example Output:

```
6
8
13
```

Problem 3: SufFIX
10 points

CODE QUEST™ 2018

Input File: `Prob03.in.txt`

Introduction

English is a strange language. We say "first," "second," and "third," but then almost every number after that ends in "th" – "fourth," "fifth," "sixth." Why can't they all be the same?

For this problem, someone has tried to "fix" a bunch of ordinal number abbreviations by changing "1st" to "1th," "2nd" to "2th," and so on, and we need you to fix them back. Solve this quickly and you may end up in firth – I mean, first – place!

Program Input

The first line of the file `Prob03.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line with a potentially incorrect ordinal number: an integer followed by the letters "th".

Example Input:

```
7
1th
2th
3th
4th
10th
12th
22th
```

Program Output

For each test case, your program should print out the number with the correct ordinal suffix. If the original number was correct, you should print it out as written.

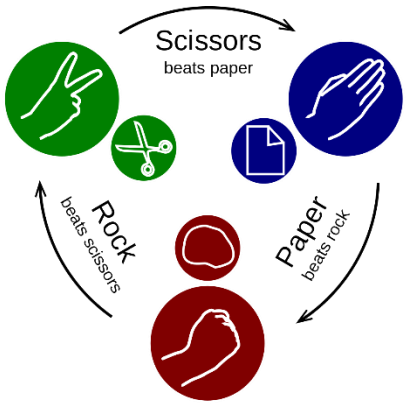
Example Output:

```
1st
2nd
3rd
4th
10th
12th
22nd
```

Problem 4: Rock, Paper, Scissors
10 points

CODE QUEST™ 2018

Input File: `Prob04.in.txt`



Introduction

Rock, Paper, Scissors is a simple game typically played head to head by two players. The players in unison say "rock, paper, scissors" and simultaneously form one of those three shapes with their hand. The winner is determined by comparing hands where ROCK beats SCISSORS, SCISSORS beats PAPER, and PAPER beats ROCK.

For this problem, you are going to be determining the winner of a series of ROCK/PAPER/SCISSORS showdowns. Simple, right? However, to make this more interesting, we are going to allow for a random number of players in each round (2 or more).

Pay close attention to the following rules when determining the winner of each round:

- ROCK beats SCISSORS
- SCISSORS beats PAPER
- PAPER beats ROCK
- If there is not a single winner, the result is "NO WINNER"

Consider these examples:

- In a 4 player game with hands of ROCK, SCISSORS, ROCK, ROCK the result is "NO WINNER" because the 3 people with ROCK each beat SCISSORS, but tie with each other.
- In a 4 player game with hands of ROCK, PAPER, ROCK, ROCK the winner is "PAPER" because the single player with PAPER beats all other players with ROCK.

Problem 4: Rock, Paper, Scissors
10 points

CODE QUEST™ 2018

Program Input

The first line of the file `Prob04.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- 2 or more letters from the set of R, P, or S (representing ROCK, PAPER or SCISSORS) in any sequence separated by single spaces.

Note: The letters (R, P and S) will always be upper case and can occur more than once in a round.

Example Input:

```
10
R S
R P R R
R P S
P S P S P R P S S R
P P P
S R S S S S S S S S S S S S S S S
P R
R S R
S P P
S S
```

Program Output

For each test case, your program should output the single winner in the form "ROCK", "PAPER" or "SCISSORS" (upper case), or "NO WINNER" if no single winner prevails.

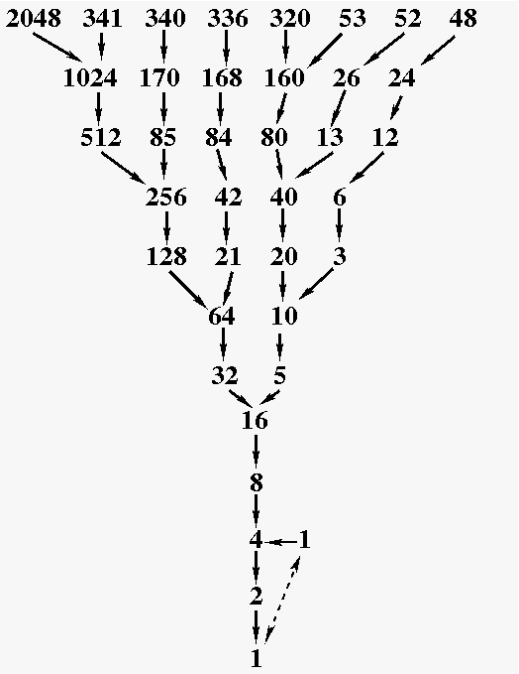
Example Output:

```
ROCK
PAPER
NO WINNER
NO WINNER
NO WINNER
ROCK
PAPER
NO WINNER
SCISSORS
NO WINNER
```

Problem 5: Collatz Conjecture
15 points

CODE QUEST 2018

Input File: `Prob05.in.txt`



Introduction

Can you believe that there are some math problems that still remain unsolved today? The Collatz Conjecture is one such problem.

To create a Collatz Sequence, start with any positive integer n. Each term in the sequence is derived from the previous term using the following rules: if the previous term is even, then the next term is one half the previous term. Otherwise, the next term is 3 times the previous term plus 1. The Collatz Conjecture states that no matter what value you pick for n, the series will eventually reach the number 1.

For example, if we start with the number 12, it is even, so the next term is 6. That is also even, so the next term is 3. Three is odd, so the term after that would be 10 and so on. The full sequence is:

12, 6, 3, 10, 5, 16, 8, 4, 2, 1

So starting with 12, the sequence length is 10. Your task is to write a program that will provide the length of a Collatz Sequence starting from a given number.

Problem 5: Collatz Conjecture
15 points

CODE QUEST™ 2018

Program Input

The first line of the file `Prob05.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single positive integer N, which will be greater than or equal to 2, and less than or equal to 1,000,000

Example Input:

```
5
12
1024
100
4
12345
```

Program Output

Your program should output the length of each sequence in the following format:

- `<Start Number>:<Sequence Length>`

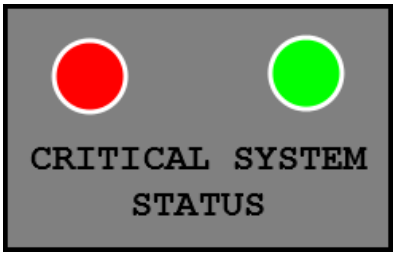
Example Output:

```
12:10
1024:11
100:26
4:3
12345:51
```

Problem 6: Space Station Repair Lights
15 points

CODE QUEST™ 2018

Input File: `Prob06.in.txt`



Introduction

You've been hired to work on a systems diagnostic unit for the International Space Station. This unit will monitor four critical systems – a Battery, a Heat Exchanger, a Water Pump, and a Temperature Sensor – to determine if they are working or not. However, space is at a premium on the ISS, and so you will only have room for two LED lights for the astronauts to see!

Your team has come up with a design that will allow the astronauts to determine which systems, if any, are broken based on the colors of the two LED lights. Each system you're monitoring is given a numerical value based on how critical it is to the operation and safety of the station:

- Battery: 8
- Heat Exchanger: 4
- Water Pump: 2
- Temperature Sensor: 1

Whenever the unit runs, it will add the values of any broken systems together and light up the LEDs to indicate the resulting number. Each LED has four states, each representing a number – off (0), red (1), green (2), and blue (3). The astronauts will multiply the left LED's value by four, then add it to the right LED's value, in order to determine the correct error code. For example, if the left LED is red and the right LED is green, the astronauts would calculate the error code as (1 * 4) + 2 = 6.

Your task is to write the logic that controls the LEDs.

Problem 6: Space Station Repair Lights
15 points

CODE QUEST™ 2018

Program Input

The first line of the file `Prob06.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line with four words separated by spaces, each either WORKING or BROKEN. These represent the statuses of the Battery, Heat Exchanger, Water Pump, and Temperature Sensor, respectively.

Example Input:

```
3
WORKING WORKING WORKING WORKING
WORKING BROKEN BROKEN WORKING
BROKEN BROKEN BROKEN BROKEN
```

Program Output

For each test case, your program should print out the correct color (in lowercase) of the two LED lights; first the left LED's color, then a space, then the right LED's color. If an LED is off, print "off."

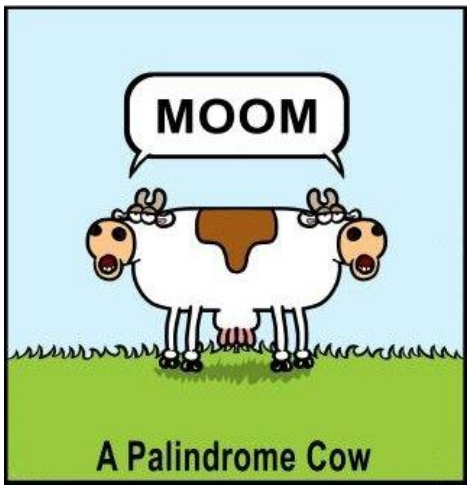
Example Output:

```
off off
red green
blue blue
```

Problem 7: Palindrome Series
15 points

CODE QUEST 2018

Input File: `Prob07.in.txt`



Introduction

A palindrome is a word, a number, or a sequence of symbols or elements which reads the same forward or backward (case is not sensitive meaning 'a' is the same as 'A').

For example, the following are palindromes:

- Madam
- Civic
- AbBa
- 123321
- $a3*3A$

You have been tasked to write a program to determine whether a given series of words, numbers, or sequences of symbols or elements are all palindromic.

Program Input

The first line of the file `Prob07.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A line containing a positive integer N denoting the number of lines that follow.
- The next N lines will contain a word, a number, or a sequence of symbols or elements.

Example Input:

```
2
4
Madam
AbBa
123321
$a3*3A$
5
Radar
aBAb
sagas
woW
12345
```

Problem 7: Palindrome Series
15 points

CODE QUEST 2018

Program Output

For each test case your program should print one of the following:

- If every item in the series is a palindrome, print: `True`
- If not every item in the series is a palindrome, print: `False – n1, n2, ...`

Where n1, n2, and so on are the non-palindromic items found in the series. The first item in the series should be number 1.

Example Output:

```
True
False - 2, 5
```

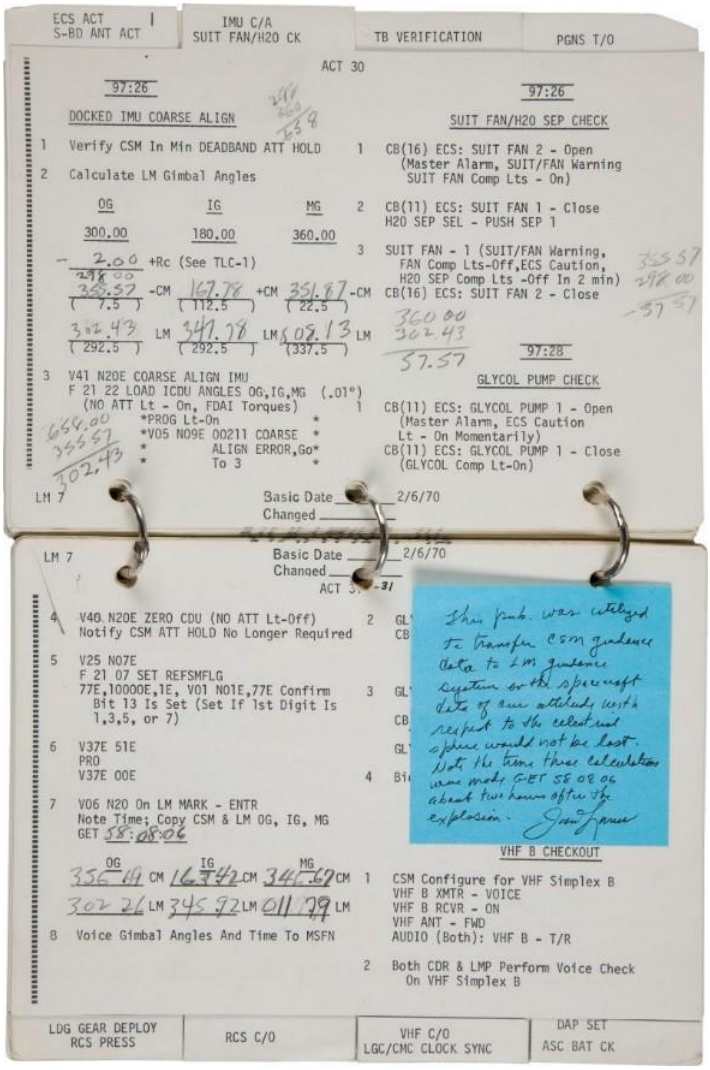
Problem 8: Apollo 13
20 points

Input File: `Prob08.in.txt`

Introduction

"Houston, we've had a problem…"

Those are words that nobody at NASA Mission Control ever wants to hear, but they were famously said on April 13th, 1970, during the Apollo 13 Mission. Just moments before, an electrical fault had caused the explosion of one of the oxygen tanks on the spacecraft carrying three astronauts to the Moon. The loss of oxygen was causing the Command Module to rapidly lose power. The only hope for survival for the three astronauts was to climb into the undamaged but much smaller Lunar Module and use it as a lifeboat to return to Earth.



Unfortunately, the Apollo 13 crew wasn't headed towards Earth; they'd already changed course to head towards the Moon. To make matters worse, the debris from the explosion had rendered their normal navigation systems useless. The Apollo crew had to work quickly with mission control to not only determine where they were in space, but get back onto a course towards Earth, and arrive before they ran out of food, water, air, or simply froze to death. Even the slightest error could have sent them hurtling off into space, never to be seen again. Within a few hours, however, they were able to develop new calculations that saw the crew of Apollo 13 safely land in the Pacific Ocean a few days later.

One of the calculations the crew had to make was intended to determine the gimbal angles between the Command and Lunar Modules. While we don't know the exact equations they used, we can simplify the problem somewhat since there aren't any lives on the line this time. If we assume that the Lunar Module and the Command Module are docked head-to-head, their gimbals should be positioned exactly opposite one another – offset by exactly 180° in all directions.

For this problem, you will need to calculate the position of the Lunar Module's gimbals on the x, y, and z axes given the position of the Command Module's gimbals and using the assumption described above.

Problem 8: Apollo 13
20 points

CODE QUEST™ 2018

Program Input

The first line of the file `Prob08.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- Three decimal numbers on one line, separated by spaces, representing the Command Module's gimbal angles on the x, y, and z axes (respectively). These values will be between 0.0 and 359.99, inclusive.

Example Input:

```
2
356.69 163.42 346.67
302.26 345.92 011.79
```

Program Output

For each test case, you should subtract 180° from each angle to provide the Lunar Module's gimbal position. Your program should output the following:

- Three decimal numbers on one line, separated by spaces, representing the Lunar Module's gimbal angles on the x, y, and z axes (respectively). These values should be between 000.00 and 359.99, inclusive; negative angles should be converted to the equivalent positive angle. Leading and trailing zeroes should be included.

Example Output:

```
176.69 343.42 166.67
122.26 165.92 191.79
```

Problem 9: Are Eucliding Me?
20 points

CODE QUEST™ 2018

Input File: Prob09.in.txt

Introduction

Coprime integers are used as values in some cryptographic algorithms such as RSA. Two integers are coprime if their greatest common divisor (GCD) is 1. There are many ways to calculate the GCD of two numbers. The Greek mathematician Euclid developed an algorithm, known today as the Euclidean algorithm or Euclid's algorithm, to calculate the GCD of two integers. There are many ways to implement this algorithm, and one way is using successive subtraction like this:

When you have a subtraction problem in the form A – B = C, the letter A is called the minuend, the letter B is called the subtrahend, and the letter C is called the difference. The following is a summary of each iteration of the algorithm:

1. Always put the bigger number as the minuend and the smaller number as the subtrahend. If they are equal, then their value is the GCD. If they are both 1, the numbers are coprime.
2. Subtract to obtain the difference between the two.
3. Use the subtrahend and the difference as the two numbers in the next iteration.

The following example shows how to get the GCD of 10 and 17:

Minuend	Subtrahend	Difference	Equation
17	10	7	17 – 10 = 7
10	7	3	10 – 7 = 3
7	3	4	7 – 3 = 4
4	3	1	4 – 3 = 1
3	1	2	3 – 1 = 2
2	1	1	2 – 1 = 1
1	1	0	1 – 1 = 0

Problem 9: Are Eucliding Me?
20 points

CODE QUEST™ 2018

Program Input

The first line of the file `Prob09.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- Two integers separated by a single comma. Both numbers will be greater than or equal to 2, and less than or equal to 1,000,000.

Example Input:

```
2
10,17
56,26
```

Program Output

Your program should output the following:

- For each iteration of Euclid's algorithm, print out the subtraction equation that results. Your last equation should have a difference of 0.
- At the end of each test case, indicate if the numbers are coprime or not by printing either COPRIME or NOT COPRIME.

Example Output:

```
17-10=7
10-7=3
7-3=4
4-3=1
3-1=2
2-1=1
1-1=0
COPRIME
56-26=30
30-26=4
26-4=22
22-4=18
18-4=14
14-4=10
10-4=6
6-4=2
4-2=2
2-2=0
NOT COPRIME
```

Problem 10: Bishop's Move
20 points

CODE QUEST™ 2018

Input File: `Prob10.in.txt`

Introduction

Chess is a well-known and well-studied game. Thousands of computer programs have been written about chess. Some play the game interactively, while others analyze and predict strategies. One thing is certain: if you don't know how the chess pieces move, your program will not be any good.

For this problem, we are going to take a look at a chess piece called the bishop. The bishop can move any number of squares, but it has to move diagonally in a straight line. Also, what fun is a standard chess board? We are going to be using boards of random size. One good thing though: your bishop will be the only piece on the board.

Program Input

The first line of the file `Prob10.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- The first line of each test case will contain two numbers, R and C, separated by a single comma that tell us how many rows and columns our chess board has. The top left square will be space (1,1), and the bottom right will be (R, C). Both R and C will be greater than or equal to 2, and less than or equal to 1,000.
- The second line of each test case will contain two numbers, r1 and c1, separated by a single comma denoting the starting row and column for our bishop.
- The last line of each test case will contain two numbers, r2 and c2, separated by a single comma denoting the ending row and column for our bishop.

Example Input:

```
3
8,8
1,1
8,8
10,20
5,6
7,8
100,100
50,50
20,21
```

Program Output

Since we are just interested in you being able to make your bishop move, your program should simply output "Yes" if the bishop can move from the starting space to the ending space in any number of moves, and "No" if it can't.

Example Output:

```
Yes
Yes
No
```

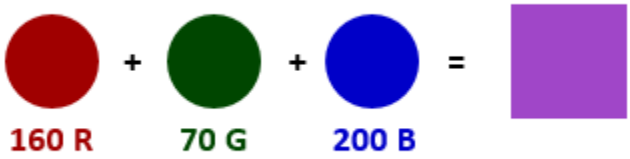
Problem 11: Chroma Key Effect
25 points

CODE QUEST 2018

Input File: `Prob11.in.txt`

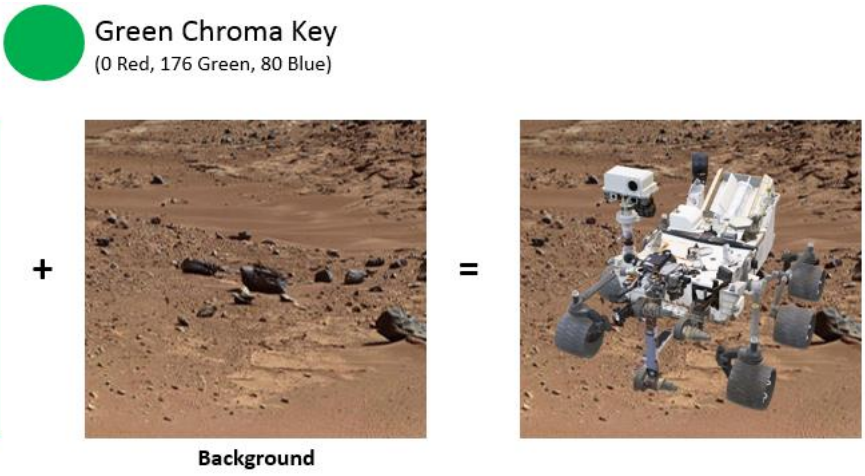
Introduction

In computer graphics, images on your screen are made up of small dots called pixels. Each of these pixels gets its color as a combination of primary colors such as Red + Green + Blue (RGB) or Cyan + Magenta + Yellow + Black (CMYK).



This might be hard to see if your copy is black and white. Red + green + blue = purple.

One common post-processing effect, used throughout the entire video industry from movies to weather, YouTube videos, live streams and more, is the application of a Chroma Key. Commonly called color keying or green screen effects, this process allows the video producer to replace part of a live video feed with an image from another video source using a colored background as a mask.

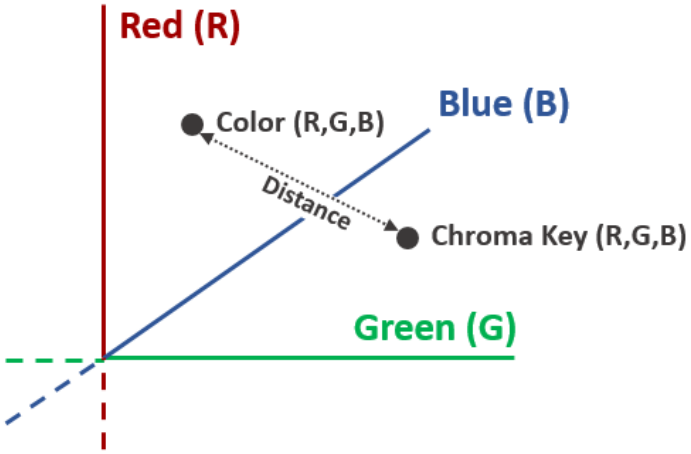


The image on the left has a green background, and the terrain is brown. The terrain replaces the green.

You've been tasked with writing software to implement chroma keying. Your application will be given the color of the chroma key (all colors are provided in RGB with values ranging from 0 to 255 [inclusive]) and a tolerance level for how "close" a color needs to be to the chroma key to be replaced. Your application will be responsible for either returning the pixel color of the foreground or the pixel color of the background depending upon the chroma key and tolerance values.

Problem 11: Chroma Key Effect
25 points

When determining whether the color is "close" enough to the chroma key, it can be helpful to consider the RGB values in a cartesian coordinate system, much like a point (x, y, z) in 3D space. If the distance between two points in this RGB space is less than or equal to the tolerance, than the colors are considered "close" and the color should be replaced by the background color.



Hint – The formula for the distance between two points will be helpful here!

$$A = (x1, y1, z1)$$

$$B = (x2, y2, z2)$$

$$Distance\ from\ A\ to\ B = \sqrt{(x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2}$$

Problem 11: Chroma Key Effect
25 points

CODE QUEST™ 2018

Program Input

The first line of the file `Prob11.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line containing 10 integers separated by spaces in the format:
  `Cr Cg Cb T Fr Fg Fb Br Bg Bb`

Notes:

- `Cr, Cg, Cb` are the Red, Green and Blue values of the chroma key
- `T` is the tolerance of the chroma key
- `Fr, Fg, Fb` are the Red, Green and Blue values of the foreground pixel
- `Br, Bg, Bb` are the Red, Green and Blue values of the background pixel

Please Note – All values are integers ranging from 0 to 255 (inclusive)

Example Input:

```
3
0 176 80 30 12 184 90 132 101 76
0 176 80 10 12 184 90 132 101 76
0 176 80 30 100 95 93 147 113 87
```

Program Output

For each test case, your program should output one line which contains three integers representing the Red, Green & Blue components of the resulting pixel.

```
Or Og Ob
```

Where `Or, Og, Ob` are the Red, Green and Blue values of the pixel result.

Example Output:

```
132 101 76
12 184 90
100 95 93
```

Problem 12: Sieve of Eratosthenes
30 points

CODE QUEST 2018

Input File: `Prob12.in.txt`

Introduction

A prime number is an integer whose only factors are one and itself. There are two generally primitive methods to determine if a number is prime. You can select an arbitrary number and test through factorization, or you can find a prime number through a number field sieve.

A Greek mathematician by the name of Eratosthenes developed an algorithm that will find all prime numbers up to a specified limit. This algorithm is known as the Sieve of Eratosthenes.

The algorithm works by starting with the first prime number, 2, and then iteratively generating composite numbers from the prime number. A composite number is a number that can be written as a product of two other numbers, neither of which is itself. These composite numbers are then "sieved out" so they no longer need to be processed, because we already know they aren't prime.

For example, let's say that we were working with the numbers up to and including 10. Our original prime candidate set would be {2, 3, 4, 5, 6, 7, 8, 9, 10}. Since 2 is first in the list, it must be prime. We can now eliminate every multiple of 2 from our prime candidates, leaving us with {3, 5, 7, 9}. Now 3 is first in the list, so 3 is a prime number. We now eliminate multiples of 3, leaving us with {5, 7}. Continuing on, we find that 5 and 7 are both primes. The final set of primes between 1 and 10 is {2, 3, 5, 7}.

Your task is to implement the Sieve of Eratosthenes, even if you can't say it!

Problem 12: Sieve of Eratosthenes
30 points

CODE QUEST™ 2018

Program Input

The first line of the file `Prob12.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A positive integer N denoting the limit for the sieve (inclusive). N will be greater than or equal to 2, and will be less than or equal to 45,000.

Example Input:

```
4
2
10
50
90
```

Program Output

Your program should output the following:

- For each prime number you find that eliminated other numbers from the list, print the size of the eliminated set in the following format:

  `Prime <P> Composite Set Size: <Size>`

- At the end of each test case, print the set of primes separated by commas and contained in curly braces.

Example Output:

```
{2}
Prime 2 Composite Set Size: 4
Prime 3 Composite Set Size: 1
{2,3,5,7}
Prime 2 Composite Set Size: 24
Prime 3 Composite Set Size: 7
Prime 5 Composite Set Size: 2
Prime 7 Composite Set Size: 1
{2,3,5,7,11,13,17,19,23,29,31,37,41,43,47}
Prime 2 Composite Set Size: 44
Prime 3 Composite Set Size: 14
Prime 5 Composite Set Size: 5
Prime 7 Composite Set Size: 2
{2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89}
```

Problem 13: Peoplebook
30 points

Input File: `Prob13.in.txt`

Introduction

In a world where technology connects people in ways that never existed before, it can be hard to keep track of everyone! Since most of your friends now have various social media accounts such as Instagram and Twitter, as well as a mobile number and email address, sometimes it's hard to remember them all.

To solve this problem, write a program that acts as a dictionary for people!

Program Input

The first line of the file `Prob13.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A line containing a number (between 2 and 20) representing the number of people for which data will be provided
- A line containing data to be stored in the dictionary. The data will be provided as an array of arrays; each array contains a different type of data and will be the same length. Each index corresponds to the same person (in other words, the first entry of each array represents the information for the first person, the second entry contains the second person's data, and so forth.). Each array, including the outer array, will be wrapped with square brackets [ ], and each entry will be separated by commas. The data contained in each array is:
    - First array: Name (Text which may contain upper- and lower-case letters and spaces)
    - Second array: Age (A number between 10 and 90)
    - Third array: Instagram username (Text which may contain upper- and lower-case letters and numbers)
    - Fourth array: Twitter handle (Text which starts with an @ symbol, then is followed by upper- and lower-case letters and numbers)
    - Fifth array: Mobile number (A ten-digit number)
    - Sixth array: Email address (Text which contains upper- and lower-case letters and numbers, followed by exactly one @ symbol, followed by lowercase letters and periods (.))
- A number of lines equal to the number of people, with each line containing the name of one person

Example Input:

```
2
2
[[Alice,Bob],[15,16],[aliceInsta,BobIsCool1],[@wonderland,@bobbyBoy],[
1234567890,4078675309],[alice123@gmail.com,bobsEmail@yahoo.com]]
Alice
Bob
2
[[Joe,Eve],[32,45],[AverageJoe,DropperOfEves],[@shoeless,@eve123],[837
4629401,3849502837],[joeabc123@orange.co.uk,eve@army.us.mil]]
Eve
Joe
```

Please note that the data is contained on a single line, but it wraps when printed out here. Check the example input file for the exact format!

Program Output

For each test case, you must print each person's profile in the order they are named after the data array. Profiles should be printed across multiple lines in the format shown below.

Example Output:

```
Name: Alice
Age: 15
Instagram: aliceInsta
Twitter: @wonderland
Phone: 1234567890
Email: alice123@gmail.com
Name: Bob
Age: 16
Instagram: BobIsCool1
Twitter: @bobbyBoy
Phone: 4078675309
Email: bobsEmail@yahoo.com
Name: Eve
Age: 45
Instagram: DropperOfEves
Twitter: @eve123
Phone: 3849502837
Email: eve@army.us.mil
Name: Joe
Age: 32
Instagram: AverageJoe
Twitter: @shoeless
Phone: 8374629401
Email: joeabc123@orange.co.uk
```

Problem 14: ZIPPER text
30 points

CODE QUEST™ 2018

Input File: `Prob14.in.txt`

Introduction

Did you know that the zipper was invented by an Electrical Engineer? Maybe that's the reason some parents say things like "you can major in whatever you want to in college, as long as it has the word engineering at the end". Anyway, the zipper has been holding together fabric and material since 1913. Today, you will tackle the zipper text cipher, which holds two different messages in one.

The zipper text cipher takes two messages – one in upper case letters, and the other in lower case letters – and zips them up into a single encoded message. Your task will be to decode such a message.

Program Input

The first line of the file `Prob14.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- The first line of each test case will contain a single integer. This is the number of lines in the upper case message (we'll call it U).
- The second line of each test case will contain U integers separated by a single space each. These are the line lengths of the upper case message.
- The third line of each test case will contain a single integer. This is the number of lines in the lower case message (we'll call it L).
- The fourth line of each test case will contain L integers separated by a single space each. These are the line lengths of the lower case message.
- The remaining lines of the file will contain the encoded message. No line in the encoded message will be longer than 80 characters. Spaces in the uppercase message will have been replaced by a minus sign, and spaces in the lowercase message will have been replaced by an equal sign. There is no punctuation in either message.

Note: because of the format of the input file, the number of test cases will always be 1!

Problem 14: ZIPPER text
30 points

Example Input:

```
1
6
35 26 28 17 21 18
5
22 26 22 23 22
I-REi=hopeA=LLyYou-=aHrOe=PE-YcOaU-LIrKefEu-THIlS-
PwitROh=mBLEinMWus=sEi-TgHnI=a
NK-IT-ISn-PREdTTY-COO=LCequalHsECKne-iTthHeE-r=ofL=tOWhER-em=cCAhSaE-
MEnSSAGEFge
iORf=y-A-oLu=cIaTTlLEl-H=tIoupNpTCODerE-cQUEaSTs-eoIS-Ar=WESOMEGtOODo-
LlowUCK-er
cEVasEe=RoYnO=theNmE
```

Program Output

Your program should output the following:

- The uppercase message, using correct line lengths and with spaces restored
- A blank line
- The lowercase message, using correct line lengths and with spaces restored

Example Output:

```
I REALLY HOPE YOU LIKE THIS PROBLEM
WE THINK IT IS PRETTY COOL
CHECK THE LOWER CASE MESSAGE
FOR A LITTLE HINT
CODE QUEST IS AWESOME
GOOD LUCK EVERYONE

i hope you are careful
with minus sign and equals
neither of them change
if you call touppercase
or tolowercase on them
```

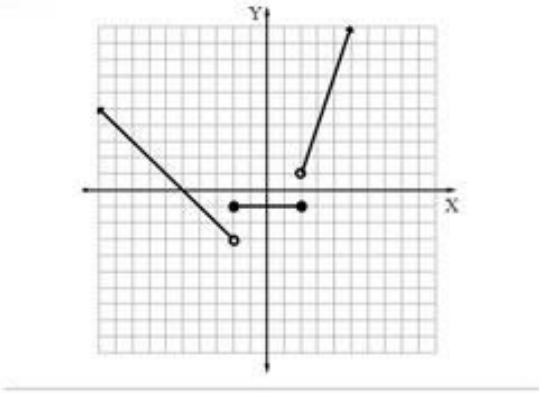
Problem 15: Piecewise Encrypter
35 points

CODE QUEST™ 2018

Input File: `Prob15.in.txt`

Introduction

In math, a piecewise function is defined by multiple sub-functions, with each sub function applying to different input values. Your task today is to create a piecewise encrypter!



$$f(x) = \begin{cases} -x - 5 & \text{if } x \le -2 \\ 3 & \text{if } -2 < x < 6 \\ x - 6 & \text{if } x \ge 6 \end{cases}$$

You were asked to create a program to encrypt a piece of text. You will be given a set of alphabetic letters to encode. Each alphabetic letter will be encoded by first being translated into a number as follows:

A=1, B=2, C=3, . . . , X=24, Y=25, Z=26

Once each letter has been converted into a number, it must be modified according to the rules in the table below.

If the letter falls within this range:	Use this encoding rule:
A – E	Add 6 to its numerical value.
F – J	Square its numerical value.
K – O	Divide its numerical value by 3. Multiply the integer remainder by 5 and add 1.
P – T	Multiply the sum of the digits of its numerical value by 8.
U- Z	Find the largest integer factor of its numerical value less than the value itself. Multiply it by 2.

Once you get the result of applying the numeric encoding rule, convert the numerical value back to an alphabetic letter to use in your output. For example, 7 would be converted into G. If the result of the rule is greater than 26, divide the number by 26 and use the remainder to determine which letter it corresponds to. For example, 27 will be encoded as A and 28 will be encoded as B. If the result of encoding calculation is 0, the original character will be used.

Problem 15: Piecewise Encrypter
35 points

CODE QUEST™ 2018

Program Input

The first line of the file `Prob15.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line of text to encode.

Example Input:

```
2
HELLO
WORLD
```

Program Output

For each test case, your program should output one line containing the encrypted text.

Example Output:

```
LKAAA
BATAJ
```

Problem 16: Twinkle Twinkle
40 points

CODE QUEST 2018

Input File: `Prob16.in.txt`

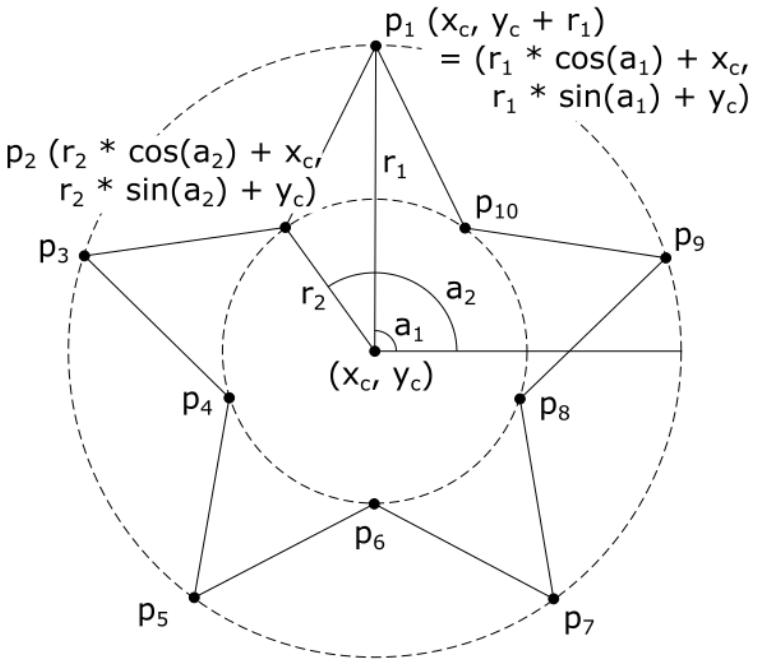
Introduction

You are working on a drawing program and you have been tasked with creating a function to draw star shapes. The program already has the capability to draw line segments between points so all you need to do is calculate the coordinates for the vertices of the star. The number of vertices in a star is double the number of the star's points, so a five-pointed star will have ten vertices: five on the tips of the star's points, and five in between the points. You can draw a circle through all the vertices on the star's points and another smaller circle through all the vertices between the points. The center point of both circles is the same as the center point of the star. The angles between the star's vertices relative to the center are the all the same and add up to 360 degrees or 2π radians. You can take advantage of these facts to calculate the coordinates of the star's vertices using the formula for the coordinates of a point on a circle:

$$x = r * \cos a + x_c$$

$$y = r * \sin a + y_c$$

where r is the radius of the circle, (x_c, y_c) are the coordinates of the center of the circle, and a is the angle between the point and the point on the circumference of the circle at zero degrees. For a point directly above the center of the circle, a will be 90 degrees or π / 2 radians.



Hint: Make sure you know if your programming language's sin and cos functions take degrees or radians and convert your angles as needed.

Program Input

The first line of the file `Prob16.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line containing five integers in the format:

  x_c y_c p r_1 r_2

  where x_c is the x-coordinate of the star's center point, y_c is the y-coordinate of the star's center point, p is the number of points in the star, r_1 is the radius of the circle that goes through the star's points and r_2 is the radius of the circle that goes through the vertices between the star's points.

Example Input:

```
3
0 0 5 100 50
50 50 8 120 70
-50 -50 4 80 40
```

Program Output

For each test case, your program should output one line. The line should contain the coordinates for each vertex in the star with a comma in between the x- and y-coordinates and a space between each vertex. The star should be oriented so that one point is directly above the center of the star at coordinates (x_c, $y_c + r_1$). The list of vertices should start with this point and continue around the star in a counter-clockwise direction listing each vertex only once. The coordinates for each vertex should be rounded to two decimal places.

Example Output:

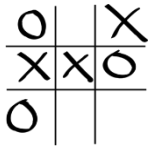
```
0.00,100.00 -29.39,40.45 -95.11,30.90 -47.55,-15.45 -58.78,-80.90
0.00,-50.00 58.78,-80.90 47.55,-15.45 95.11,30.90 29.39,40.45

50.00,170.00 23.21,114.67 -34.85,134.85 -14.67,76.79 -70.00,50.00 -
14.67,23.21 -34.85,-34.85 23.21,-14.67 50.00,-70.00 76.79,-14.67
134.85,-34.85 114.67,23.21 170.00,50.00 114.67,76.79 134.85,134.85
76.79,114.67

-50.00,30.00 -78.28,-21.72 -130.00,-50.00 -78.28,-78.28 -50.00,-130.00
-21.72,-78.28 30.00,-50.00 -21.72,-21.72
```

Note: Because these lines are long, we have added a blank line between test case outputs only in the problem packet. Do not do this in your program! Compare your output to the example output file provided!

Problem 17: Tic Tac Toe Checker
45 points

CODE QUEST 2018

Input File: `Prob17.in.txt`



Introduction

Tic-tac-toe is a game for two players, X and O, who take turns marking the spaces in a 3 x 3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical or diagonal row wins the game.

The following example game is won by the first player, X:

You have been hired by CoolGames, Inc. to create a tic-tac-toe checker program that they can use to determine whether X wins, O wins, or if a tie has resulted.

Program Input

The first line of the file `Prob17.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line of characters denoting the current state of a Tic-Tac-Toe game

The game state will represented as follows:
<1 VALUE><2 VALUE><3 VALUE><4 VALUE><5 VALUE><6 VALUE><7 VALUE><8 VALUE><9 VALUE>

1	2	3
4	5	6
7	8	9

Figure 1. Tic-Tac-Toe Mapping

0	-	X
-	O	O
X	X	X

Figure 2. Tic-Tac-Toe Game

123456789
O-X-OOXXX

Figure 3. Tic-Tac-Toe Encoding

In Figure 1, it shows how the values in the tic-tac-toe board are mapped. In Figure 2, it shows an example tic-tac-toe game being played where "-" means no one has played in the area. In Figure 3, it shows how the tic-tac-toe game is encoded into a string of values based on the board mapping.

Example Input:

```
8
O-X-OOXXX
XOX-OXO-X
X-O-XO--O
OXOXXOXOX
--X-X-XOO
XXOXO-O--
XOXXOOOXX
---------
```

Program Output

For each test case, your program should output one line in the following format:

- The original input line followed by a single space, an equal sign, and another single space
- One of the following three phrases depending on the outcome of the game:
  - X WINS
  - O WINS
  - TIE

Example Output:

```
O-X-OOXXX = X WINS
XOX-OXO-X = X WINS
X-O-XO--O = O WINS
OXOXXOXOX = TIE
--X-X-XOO = X WINS
XXOXO-O-- = O WINS
XOXXOOOXX = TIE
--------- = TIE
```

Problem 18: Dominating Disney
60 points

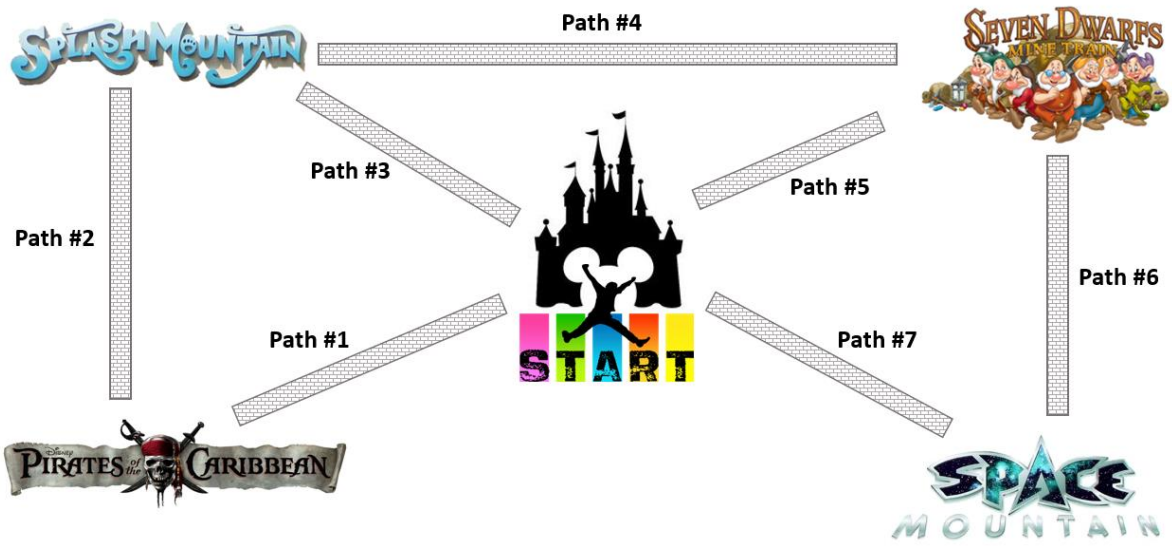
CODE QUEST 2018

Input File: `Prob18.in.txt`

Introduction

Your family is going to Disney World in Orlando, FL during the busiest time of the year, so it's up to you to plan the most efficient way to hit the most popular rides: Pirates of the Caribbean, Seven Dwarfs Mine Train, Space Mountain, and Splash Mountain. Your challenge is to find the quickest way to navigate through crowds, parked strollers, and character Meet-and-Greets.

The map below shows the locations of all four rides and the paths that lead to them. You will be given the amount of time it takes to walk each path. If there is ever a tie between two or more paths, choose the path that starts with the smallest Path #. For example, if Path #1 > Path #2 > Path #4 > Path #6 takes the same amount of time as Path #7 > Path #6 > Path #4 > Path #2, you should choose the first option. Sometimes it may be fastest to revisit a ride to which you've already been!



Program Input

The first line of the file `Prob18.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line with seven positive integers separated by spaces, corresponding to the time it will take to walk each of the seven paths in the map above. The path times will be in order, starting with Path #1's time and ending with Path #7's. All path times will be greater than 0.

Example Input:

```
3
5 10 7 6 2 4 12
15 8 20 7 20 5 10
4 8 10 6 6 8 10
```

Program Output

For each test case, your program should output one line containing the fastest path to ride all four attractions. If the fastest path is through Path #1 > Path #2 > Path #3 > Path #5 > Path #6, then you will print: `1 2 3 5 6`

Example Output:

```
1 2 4 6
7 6 4 2
1 2 4 6
```

Problem 19: It's an Enigma!
70 points

CODE QUEST™ 2018

Input File: `Prob19.in.txt`

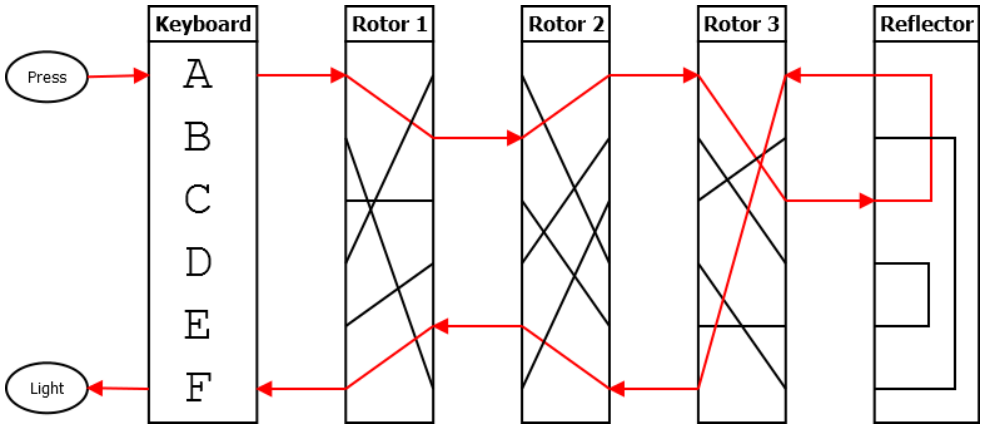
Introduction

During World War II, the Germans used the Enigma cipher to encrypt most of their military communications. Thought to be unbreakable, the cipher made use of typewriter-like machines that included a complicated system of rotors to scramble the text of a message. Which rotors were used, in which order, and in what positions they were originally set determined the key used to encrypt and decrypt an Enigma message. This complicated system led to trillions of possible keys and foiled traditional methods of breaking the cipher.

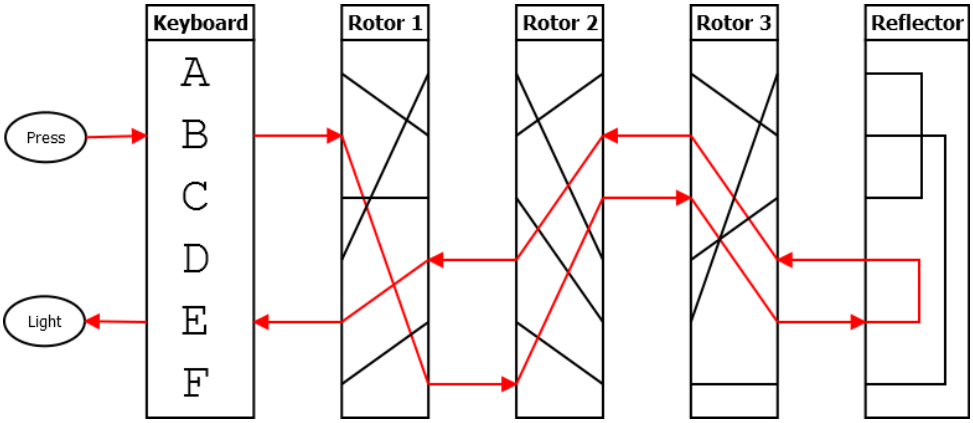
To use an Enigma machine, an officer would press a button on a keyboard. This would send an electrical signal through the machine and into the rotors. Each rotor would have an input and output connection for each letter of the alphabet, but the signals would be scrambled; a signal going into the "A" position may come out at the "E" position instead. Each rotor was wired differently and numbered so they could be easily identified. Once the signal made it through all three rotors, it would go through a reflector which would scramble the letters again and send the signal back through the rotors one more time. The signal would then finally turn on a light showing which letter it had turned into. The officer would write down this letter, then continue to the next letter.

To ensure the cipher's security, the rotors would rotate while the message was entered. After each letter was pressed, the rightmost rotor would rotate one position. When that rotor returned to its original position (after 26 letters), the middle rotor would rotate one position. Once the right rotor completed another full rotation, the middle rotor would rotate one more position. Once the middle rotor had completed a full rotation in this manner, the left rotor would rotate one position.

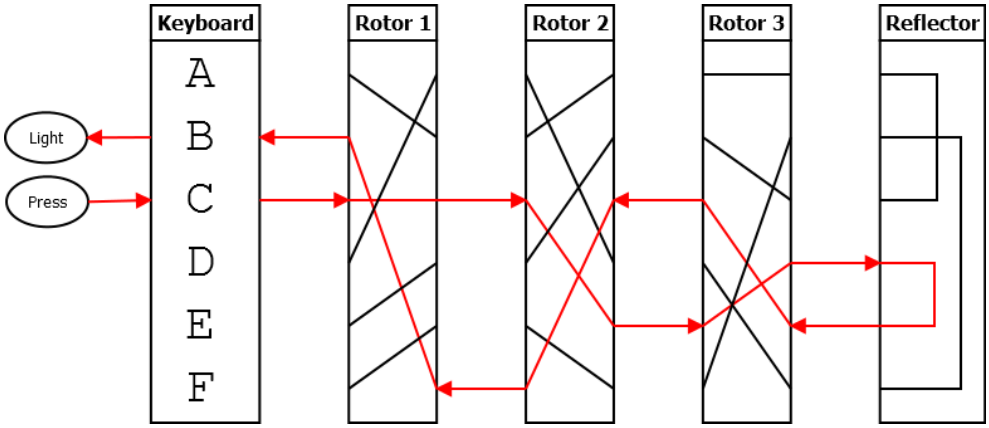
The diagrams below show what happens when trying to encrypt the message "ABC" using a six-letter Enigma machine. The first letter, A, becomes B after leaving the left rotor, an A after leaving the middle rotor, and then a C after leaving the right rotor. It enters the reflector, becoming an A again, then goes back through the rotors in reverse order. It then becomes an F, then an E, then an F again as it leaves the left rotor. The light for F turns on, indicating that the original letter A should be encoded as an F.



The right-most rotor then shifts down one position. This causes B to be encoded as E.

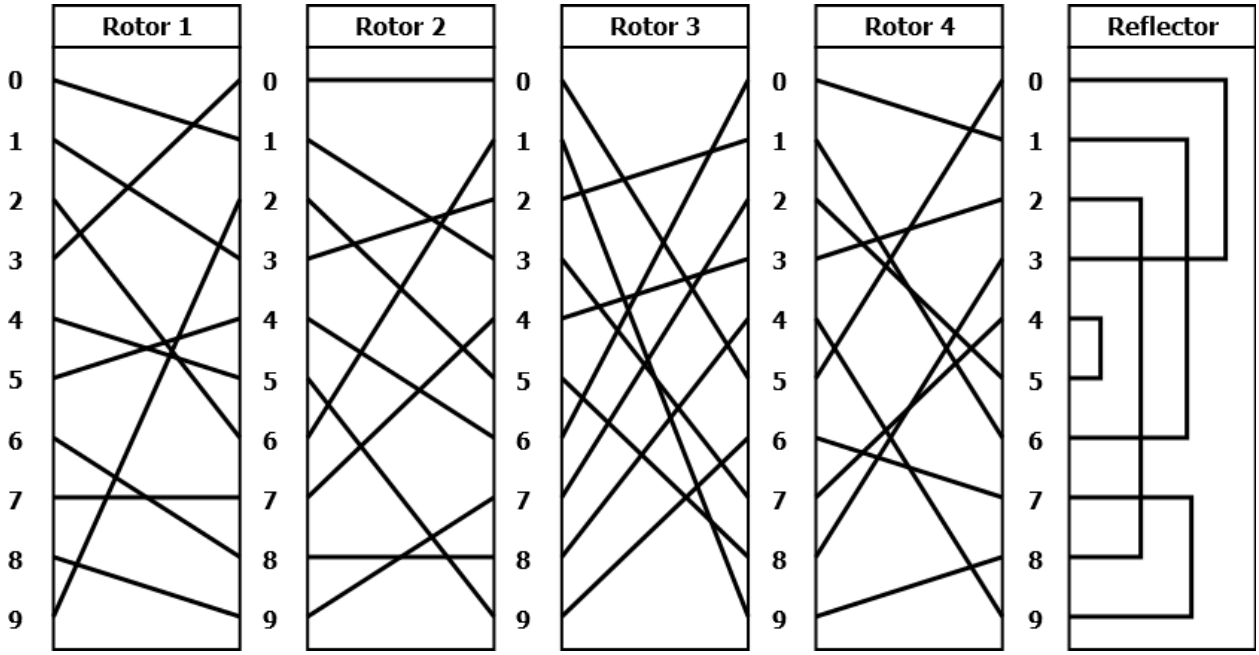


The right rotor shifts once more, and the third letter, C, gets encoded as B. The final message is "FEB."



You have been tasked with creating a digital version of the Enigma cipher. Rather than encoding letters of the alphabet, it will encode the digits 0 through 9. Four rotors and a reflector are pictured below. However, only three of these rotors (and the reflector) will be used in each problem. The inputs provided will indicate which rotors are to be used in which positions. The layout of the rotors and reflector is shown below. You have three of each rotor, so rotors can be used more than once to form a sequence of three rotors. The reflector is always in the fourth position.

Problem 19: It's an Enigma!
70 points



Each rotor above is shown in starting position 0; however, each key will require that the rotors start in a different position. A rotor in starting position 1 should begin rotated down one position as shown in the examples above; a rotor at starting position 2 should begin rotated down two positions, and so on. The reflector does not rotate and will remain in the position as pictured above.

Problem 19: It's an Enigma!
70 points

CODE QUEST™ 2018

Program Input

The first line of the file `Prob19.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A line containing a rotor number (1-4), a space, and that rotor's starting position (0-9), representing the left rotor
- A line containing a rotor number (1-4), a space, and that rotor's starting position (0-9), representing the middle rotor
- A line containing a rotor number (1-4), a space, and that rotor's starting position (0-9), representing the right rotor
- A line containing a number to be encrypted.

Example Input:

```
2
1 0
2 0
3 0
1234567890
4 4
3 1
1 7
489410751741041749171971079412 30
```

Program Output

Your program should print out the encrypted numbers. Leading zeros must be included.

Example Output:

```
4805344463
150657449303555668500006606030 93
```

Rounding

For all Code Quest problems that ask you to round, we will be using the "round to nearest" method, which is what most people consider to be normal rounding. If we were asked to round to the nearest integer, the following results would occur:

- 1.49 would round down to 1
- 1.51 would round up to 2

Because we are rounding to the nearest item, what happens when a number is exactly in the middle? In that case, we will use the "round away from zero" tie breaker, which is also what is generally considered to be normal rounding. Again, if we were rounding to the nearest integer:

- 1.5 would round up to 2
- -1.5 would round down to -2

You should use this method of rounding unless a problem explicitly tells you do use another specific type of rounding.