1. The Bases Are Covered

Program Name: Bases.java Input File: bases.dat

Dr. Long is a Computer Science teacher and needs a table of numbers in different bases. Given a positive integer `n`, he wants all of the numbers from 1 to `n` printed in a table with the base 10 number first followed by base 2, base 4, base 8, base 12 and base 16. You have been assigned to write this program for him.

Input
The only line of input will contain a single integer $n \leq 45$ that indicates the last base 10 row to be printed.

Output
In a table with headers and format as shown below, you will print the base 10, 2, 4, 8, 12, and 16 values. Use lower case letters for Base 12 and Base 16 values greater than 9, as shown below.

Note: The width of the columns is not important as long as the data in the columns are left justified and there is at least one space between any two numbers in a row.

Example Input File
```
13
```

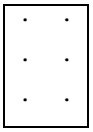
Example Output to Screen
```
Base 10   Base 2    Base 4    Base 8    Base 12   Base 16
1         1         1         1         1         1
2         10        2         2         2         2
3         11        3         3         3         3
4         100       10        4         4         4
5         101       11        5         5         5
6         110       12        6         6         6
7         111       13        7         7         7
8         1000      20        10        8         8
9         1001      21        11        9         9
10        1010      22        12        a         a
11        1011      23        13        b         b
12        1100      30        14        10        c
13        1101      31        15        11        d
```

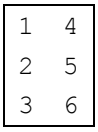
2. Braille Student

Program Name: Braille.java Input File: braille.dat

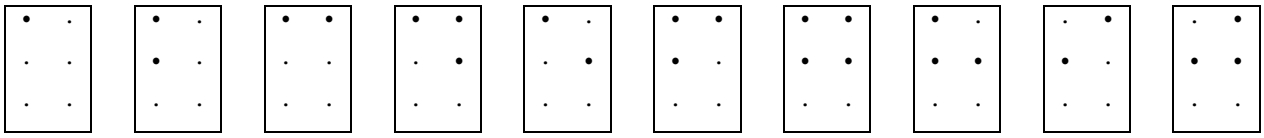
In the early 1800's, Louis Braille invented what is now called the Braille system that many visually impaired students use today. Aaron is a new student at your school and has to have his papers transcribed into Braille so he can read them. You have decided to write a program that will read a stream of Braille characters and print the corresponding letters or numbers.

Braille characters are formed by a matrix of dots in a cell with 3 rows and 2 columns. The dot in each of the six positions of a cell is either raised (bold) or flat (not-bold). A diagram of the dot positions is at the right.





The matrix of dots in a cell is used to distinguish between letters, digits, and special characters such as punctuation marks and diacritics. For instance, the following cells can represent the letters a-j or the digits 1-0 as described below.



a or 1 b or 2 c or 3 d or 4 e or 5 f or 6 g or 7 h or 8 i or 9 j or 0

All cells represent a space, a punctuation mark, a letter, a digit, or a "sign" to switch between upper and lower case letters and digits using the following Braille rules:

- At the beginning of a message, the default interpretation is a lower case letter.
  - The cells in the diagrams shown above represent the letters a-j.
  - Letters k-t are represented a dot in position 3 plus the code for letters a-j. For example, a cell representing the letter k would have dots 1 and 3 raised.
  - Letters u, v, x, y, and z are represented by dots in positions 3 and 6 plus the code for a, b, c, d, and e respectively.
  - There was no letter w in the French alphabet that Braille used but has since been added as 3 and 6 plus the code for j.
- A "sign" is used to switch between upper and lower case letters and digits. There is one number sign, and three letter signs:
  - **Number sign:** a cell that contains the dots 3, 4, 5, and 6. All cells following a number sign represent single digits, 1 through 0 as shown above, until a space or one of the three letter signs below is encountered.
  - **Letter signs:**
    - **letter sign** is a cell that contains the dots 5 and 6. The cells continue to represent lower case letters until a different sign is encountered.
    - **capital letter sign** is a cell that contains only a dot in location 6. Only the letter in the cell following the capital letter sign is capitalized and the remaining cells represent lower case letters until a different sign is encountered.
    - **all capital letter sign** is represented by two consecutive capital letter signs. All letters will be capitalized until a different sign is encountered.
- A space is represented by a blank cell.
- The only punctuation mark for this problem is a period, represented by a cell containing dots 2, 5, and 6.

Input

The first line of input will contain a single integer `n` that indicates the number of Braille phrases to follow. Each of the Braille phrases will consist of contiguous cells each of which contains 3 rows and 2 columns of ones and zeroes and represent Braille code as described above. A one represents a dot and a zero represents no dot.

(continued on next page)

2. Braille Student (cont.)

Output
You will print the English phrase represented by the Braille code.

Example Input File
```
2
100011101100101001000110000011101010010 0
000001011100110010000101000010100100101 1
000000100000000001000110000010010000010 01
001110111110011010000000011101101111 10
000001001000110111000000100010010100 01
010010101011100010000101100000000100000
```

Example Output to Screen
```
a dog has 5 Fleas.
Computer SCIENCE
```

3. Chutes

Program Name: Chutes.java Input File: chutes.dat

You have been contacted by a game company to write a game engine that can play Chutes and Ladders for any arbitrary 8 by 8 game board. The path along the board from 1 to 64 is shown in the grid to the right.

64	63	62	61	60	59	58	57
49	50	51	52	53	54	55	56
48	47	46	45	44	43	42	41
33	34	35	36	37	38	39	40
32	31	30	29	28	27	26	25
17	18	19	20	21	22	23	24
16	15	14	13	12	11	10	9
1	2	3	4	5	6	7	8

The rules of the game are pretty simple:

- All players begin play in square one, the lower left corner of the grid.
- Each player then, in order, takes a turn rolling a 6 sided die with the numbers 1 through 6 on the sides.
- The player then, beginning from square number one, advances his token the number of squares that corresponds to the number he rolled on the die.

After a player has moved the number of spaces that are indicated on the die, he will have moved to a space that has exactly one of three possibilities:

- If a player lands on a space that contains the bottom of a ladder (referred to as "landing on a ladder"), he moves up the ladder to a new space as described below.
- If he lands on a space with the top of a chute (referred to as "landing on a chute"), he moves down the chute to a new space as described below.
- If the new space he lands on is neither a chute nor a ladder, he stays at that position until his next turn.

Other rules are:

- If case 1 or case 2 above moves him to another ladder or chute, he performs the same action as listed above, and continues doing so until he gets to a space that is neither a chute nor a ladder. At that point in the game, he is in state 3 above and he stays at that position until his next turn.
- Players rotate taking their turns beginning with player A as described below.
- The game is won when a player rolls a number that will move him to (or past) the box labeled 64.
- There is guaranteed to not be a chute in 64, as that would prevent the game from being solvable.
- There will also never be a chute or ladder starting from 1.

To simulate the dice rolls, you should construct an object of the type `java.util.Random`. This class allows you to specify the seed for the random number generator. For a given seed, the order of the random numbers is always the same. You should also use the `nextInt` method to generate the rolls.

Input

The first line will contain a single integer `n` that indicates the number of games to follow. For each game,

- The first line contains a single integer `p` that indicates the number of players in the game.
- The next line will contain the seed, a `long` integer.
- The following line will contain 2 integers `c` and `d` separated by a space. The first integer `c` is the number of chutes; the second `d` is the number of ladders.
    - Next are `c` lines containing an integer pair for each chute; the first integer is where a chute begins and the second is where the player ends up at the end of the chute.
    - Then there are `d` lines containing an integer pair for each ladder; the first integer is where the ladder begins, and the second is where the player ends up at the top of the ladder.
    - The integer values for the chute and ladder pairs correspond to the values in the grid above.

(continued on next page)

3. Chutes (cont.)

Output

Each player in a game is given an uppercase character, with the first player being `A`, the second being `B`, etc. For each game you should output a single line that reads `Player X wins after Y rolls!` where *X* stands for the winner of the game, and *Y* is the total number of rolls that happened overall in the game, including the winning roll. If for example there are two players, and player `A` rolled 6 times and player `B` rolled 5, and player `A`'s sixth roll won the game, then the string would read `Player A wins after 11 rolls!`

Example Input File

```
2
2
3735928559
4 4
27 10
10 6
57 41
34 14
14 37
17 49
50 64
22 37
3
4206243583
3 4
61 45
60 45
45 37
17 48
4 54
11 39
40 56
```

Note: these are the random numbers generated until the final roll for each seed in the input file:

3735928559

```
2 3 5 6 4 2 6 4 4 4 5 4 2 4 3 5 2 4 4 2 5 1 4 5 5
```

4206243583

```
2 2 5 6 2 2 1 2 6 5 4 3 6 4 6 4 1 4 1 5 3 1 2 4 2 6 6 2 5 6 6 3
```

Example Output to Screen

```
Player A wins after 25 rolls!
Player B wins after 32 rolls!
```

4. Emergency Room

Program Name: Emergency.java Input File: emergency.dat

A particular hospital emergency room operates as follows:

1. When patients arrive they are immediately evaluated and given a severity score, 1 through 10, with the higher numbers being more severe. Assume the evaluation doesn't take any time.
2. Then the patient is asked to fill out paperwork that takes 5 minutes. Patients with severity 8 or higher can be seen before the paperwork is complete, and will finish the paperwork after the doctor is done. Paperwork completed after seeing the doctor does not count toward the wait time.
3. When a doctor is ready for a new patient, he grabs the highest severity person that has also been waiting the longest.
4. The wait time starts when the patient arrives at the ER, includes the time spent completing the paperwork (as described in item #2 above) before seeing the doctor, and ends when the doctor starts seeing the patient.
5. Assume that for any given severity, the doctor will spend the severity number times 8 minutes with the patient.
6. The hospital will always have a minimum of 3 doctors on duty during the 24 hour period.

You have been hired by the hospital to create a simulation to help them determine the optimal number of doctors that need to be on duty during a given 24 hour period so that the average wait time for all patients that enter the ER during that period is less than or equal to a given target wait time. You are to use the ER operations listed above to create the simulation. The hospital has given you several scenarios (described in the Input section below) to test your simulation.

Input

The first line will contain a single integer `n` that indicates the number of simulation scenarios to follow. For each scenario:

- The first line contains two integers e and a, separated by a space, where e is the number of entries in the simulation and `a` (`a` $\geq$ `5`) is the target maximum wait time for a patient to be seen by a doctor, in minutes..
- The next `e` lines contain a time in 24 hour format, followed by a space and an integer `s` ($1 \leq$ `s` ≤ 10) denoting the severity of the patient.
- The entries in a scenario will be in the order patients arrive to the ER for a 24 hour period starting at 00:00 and ending at 23:59.

Output

For each scenario, you will print the minimum number of doctors that need to be on duty so the average wait time for the patients is less than or equal to the target wait time goal, followed by a space and the word `doctors`.

Example Input File

```
1
8 5
00:01 9
00:01 9
00:01 3
00:06 8
11:01 3
11:15 8
11:30 8
23:10 1
```

Example Output to Screen

```
4 doctors
```

5. Etch-a-Sketch

Program Name: Etch.java Input File: etch.dat

Ricardo has been playing with his little brother's travel sized Etch-a-Sketch for a while. A travel sized Etch-a-Sketch has a film in a 25 unit long by 15 unit tall rectangular frame that has a moveable stylus. There are two knobs that control the movement of the stylus. The left knob moves the stylus left or right and the right knob moves the stylus up or down. Ricardo has decided that it would be fun to have a program that would simulate the movement of the stylus in an Etch-a-Sketch frame. You have agreed to write a simple simulator for him.

The simulator will draw a picture by reading a series of commands from a data file. Each command is a composed of a single letter `L`, `R`, `U`, or `D` (for left, right, up, or down respectively) that denotes the direction of the move and a single integer denoting how many units the stylus will move in the given direction. For your purposes, the upper left corner of the frame has coordinates 1, 1. If the command given would move the stylus off the frame, the stylus just stops at the edge of the frame and continues from there for the next instruction.

Input

The first line of input will contain a single integer `n` that indicates the number of pictures that will be drawn. For each picture:

- The first line will contain an ordered pair `r c` denoting the row and column respectively of the initial location of the stylus relative to the upper left corner, `1 1`. That location is considered to be drawn.
- The second line will contain a series of moves, separated by a space, in the form *dx* where *d* is the direction (`L`, `R`, `U`, or `D`) and *x* is the distance the stylus is to be moved.

Output

For each picture, you will print the 15 x 25 unit matrix that is the result of all of the moves in a given picture. A period (`.`) is a point that was not drawn and an asterisk (`*`) is a point that was drawn. Print at least one blank line after each picture.

Example Input File

```
1
5 5
R10 D10 L5 U5 R10 D10 U5 L3 U5 R4 U13 L22 D5 R4
```

Example Output to Screen

```
**********************....
*.....................*....
*.....................*....
*.....................*....
*...**********.*****.....
*****.........*.*.........
..............*.*.........
..............*.*.........
..............*.*.........
.........***********......
.........*....*....*......
.........*....*....*......
.........*....*....*......
.........*....*....*......
.........******....*......
```

6. Message

Program Name: Message.java Input File: message.dat

The police have been investigating some crimes involving a serial robber. After each robbery, the robber has left a note at the scene that has been meaningless. However, one detective has a theory that if the words of a certain length are extracted from the note and listed in the order that they appear, the note will give a clue to the identity of the suspect.

You are to write a program that will find the message for him.

Input

The first line of input will contain a single integer `n` that indicates the number of notes to follow. Each the following notes will consist of two lines.

- The first line will be a single integer `m` that denotes the length of the words that you will extract from the note.
- The second line will be a single paragraph that contains one note written in all uppercase letters, spaces, periods and commas.

Output

In the order that the words appear in the note, you will print the words of length `m`. There should be a space after each word but do not include any punctuation.

Example Input File

```
2
4
MERRY EASTER TO EVERY ONE NAME LICKETY SPLIT IS A HAPPY WORLD MIKE MAY HAVETO
CONVEY A LIVELY WATERING HOLES TO LIVE IN A PINKISH RIBBON OF CANDY. TOM ROSS
HAS A LOVELY, WHITE HOUSE ON THE END OF THE MEADOW ON LINDSEY LANE IN
BEAUMONT, TEXAS.
5
RICHARD CAN HAVE A VERY LONG PARIS TRIP TO THE END OF THE BORDER BUT BETTER
HOMES CAN BE FOUNDED ON THE WATER. THE NICEST BANKS IN THE LAND ARE NEAR THE
EDGE OF THE RED RIVERR. THE BUSHES ARE NEXTT TO THE WEDES IN THE MIDDLE OF
THE LAKE. I CAN SEEYA HAVE A LOT OF BUSINESS TO DO SO PLEASE ENJOYY
DECIPHERING THIS NOTE.
```

Note: Although the paragraphs in example input above appear on the printed page to consist of more than one line, each one is actually a single line in the input file.

Example Output to Screen

```
NAME MIKE LIVE ROSS LANE
PARIS HOMES WATER BANKS NEXTT WEDES SEEYA
```

Note: An extraneous space at the end of each line of output is ok.

7. Most and Least

Program Name: MostLeast.java Input File: mostleast.dat

Ms. Appleworth is a teacher and needs you to write a program that will find the word or words that are repeated the most number of times in a paragraph and the word or words that are used the least in the paragraph.

Input

The first line of input will contain a single integer `n` that indicates the number of paragraphs to follow. Each of the following `n` lines will contain a single paragraph.

- All of the words in the paragraph will be separated by a space, a period, a comma, or a question mark.
- Any other characters in the word, such as an apostrophe or hyphen are considered to be part of the word.
- Each paragraph will be less than 100 words long and word(s) that repeat the most will not be the word(s) that appear the least number of times.

Output

For each paragraph, you will have two lines of output.

- The first line will contain the number of times the word or words that are most repeated appear in the paragraph followed by an alphabetical list of those words, each followed by a space.
- The second line will contain the number of times the word or words that are used least in the paragraph were used followed by an alphabetical list of those words, each followed by a space.

Example Input File

```
2
MARY HAD A LITTLE LAMB, LITTLE LAMB, LITTLE LAMB, MARY HAD A LITTLE LAMB, ITS
FLEECE WAS WHITE AS SNOW. AND EVERYWHERE THAT MARY WENT, MARY WENT, MARY
WENT, AND EVERYWHERE THAT MARY WENT, THE LAMB WAS SURE TO GO.
BAA, BAA, BLACK SHEEP, HAVE YOU ANY WOOL? YES SIR, YES SIR, THREE BAGS FULL.
ONE FOR THE MASTER, AND ONE FOR THE DAME, AND ONE FOR THE LITTLE BOY WHO
LIVES DOWN THE LANE. ONE FOR THE MASTER, AND ONE FOR THE DAME, AND ONE FOR
THE LITTLE BOY WHO LIVES DOWN THE LANE.
```

Note: Although the paragraphs in example input above appear on the printed page to consist of more than one line, each one is actually a single line in the input file.

Example Output to Screen

```
6 MARY
1 AS FLEECE GO ITS SNOW SURE THE TO WHITE
8 THE
1 ANY BAGS BLACK FULL HAVE SHEEP THREE WOOL YOU
```

Note: An extraneous space at the end of the list of words is ok.

8. Petite Pals

Program Name: PetitePals.java Input File: petitepals.dat

There are many words that are palindromes, that is, the word reads the same when the letters are reversed. The word `RADAR` is an example of a palindrome.

Some words are what we will call Petite Palindromes or Petite Pals for short. Petite Pals are substrings of a given word that are palindromes that are no longer than the length of the original word –1. For instance, for `RADAR`, the Petite Pals are `ADA`, `R`, `A` and `D`.

You are to write a program to determine how many distinct Petite Pals there are in a given word.

Input
The first line of input will contain a single integer `n` that indicates the number of words to follow. Each of the following `n` lines will contain a single word consisting of at least two uppercase letters.

Output
For each word input, you will print the number of Petite Pals contained in that word.

Example Input File
```
3
RADAR
STATEROOM
ATOYOTA
```

Example Output to Screen
```
4
9
6
```

9. Rasterizer

Program Name: Rasterizer.java Input File: rasterizer.dat

The company you work for is in need of a simple rasterizer for a drawing program for kids. Since your boss is so impressed with your programming skills, he has asked you to implement it. A rasterizer is an algorithm that takes drawing commands and renders the 2D image they describe. Each draw command is comprised of:

- a primitive keyword indicating the shape,
- the color used, and
- the primitive specific details, such as the location, width and height for a square or the starting and ending points for a line.

Before the code goes into the application your boss wants you to write a small program to test it. You will read information from a text file and output the image as text. For simplicity the colors will just be uppercase characters A to Z. Each line will contain a single primitive keyword along with the necessary details to rasterize it as shown in the table below.

The primitive line formats you will need to support are:

Primitive Keyword and Details	Description
BOX color tlx tly brx bry fill	color: the color character tlx: integer for the x coordinate of the top left corner tly: integer for the y coordinate of the top left corner brx: integer for the x coordinate of the bottom right corner bry: integer for the y coordinate of the bottom right corner fill: `Y` if you should fill the whole box with the color, `N` if you should just draw the 1 pixel outline
LINE color x1 y1 x2 y2	color: the color character x1: integer for the x coordinate of the starting point y1: integer for the y coordinate of the starting point x2: integer for the x coordinate of the ending point y2: integer for the y coordinate of the ending point **NOTES:** Either x1 and x2 will be equal, or y1 and y2 will be equal (no diagonal lines)
CROSS color cx cy w h	color: the color character cx: integer for the x coordinate of the center of the cross cy: integer for the y coordinate of the center of the cross w: total width of the cross h: total height of the cross **NOTES:** If the width or height is even, add the extra pixel to the right or the bottom (towards the maximum)

(continued on next page)

9. Rasterizer (cont.)

Input

The first line will contain a single integer `n` that indicates the number of image descriptions to follow. For each image description:

- The first line contains 2 integers `w` and `h` indicating the width and height of the image.
- In rendering, screen resolution, position and size are done in width x height order, that is, the `x,y` coordinate of the top left of the image is always `0,0`, with `x` increasing positively to the right to `width-1` and `y` increasing downwards to `height-1`.
- Initially, the image should have all cells initialized to the period character '`.`' to denote that nothing has been drawn there.
- Read and process the commands one line at a time:
    - For the primitive keywords `BOX`, `LINE`, or `CROSS` found at the beginning of each input line, use the description from the table on the previous page corresponding to each keyword to interpret the details following the keyword on the input line. For instance, for the input line `BOX W 1 1 3 4 Y` the drawing program should draw a box from `1,1` to `3,4` filling the whole box with `W`.
    - Use the single word `END` to complete the current image by printing it.

Output

For each image you will output the image, a row per line, without any spaces between columns. There should be a blank line between images.

Example Input File

```
1
10 8
BOX W 1 1 3 4 Y
BOX A 9 7 100 100 N
LINE L 2 4 8 4
CROSS X 5 4 4 4
END
```

Example Output to Screen

```
..........
.WWW......
.WWW......
.WWW.X....
.WLLXXXXL.
.....X....
.....X....
.........A
```

10. Remnants

Program Name: Remnants.java Input File: remnants.dat

A local big box store sells garden hose remnants in 14 foot lengths. Many customers need the salesman to tell them how many remnants they will have to buy given the number of yards they need. Since the salesmen were frequently incorrect with their conversions thus making the customers unhappy, the company wants you to write a program that will do this conversion for the salesman.

Input
The only line of input will contain an unknown number of positive integers, but fewer than 100, that each represent the number of yards of garden hose that a customer needs. Each of items will be separated by a space.

Output
For each integer that is input, you will print on a separate line the number of garden hose remnants that the customer will need to buy.

Note: There are three feet in a yard.

Example Input File
```
7 120
```

Example Output to Screen
```
2
26
```

11. Spiral Galaxies

Program Name: Spiral.java Input File: spiral.dat

As an astronomer, Danielle has studied spiral galaxies for years. Spiral galaxies consist of a flat, rotating disk containing stars, gas, and dust around a central concentration of stars known as the bulge thought to host a supermassive black hole at its center.

A spiral galaxy is so-named because there are more young stars, which are brighter than old stars, that die out quickly leaving the remaining brighter stars standing out against a darker background. These waves of stars are more visible and appear to form a spiral, called spiral arms, within the galaxy, hence the name spiral galaxy.

Some astronomers refer to the location of a given star relative to the bulge where the bulge is at location 1 and the stars spiral counter-clockwise around the bulge as shown to the right in the order indicated by the numbers in black. The number 1 represents the location of the bulge and locations beginning with 2 represent the possible locations of the bright, young stars in the spiral arms of the galaxy.

	1	2	3	4	5
1	**13**	**12**	**11**	**10**	**25**
2	**14**	**3**	**2**	**9**	**24**
3	**15**	**4**	**1**	**8**	**23**
4	**16**	**5**	**6**	**7**	**22**
5	**17**	**18**	**19**	**20**	**21**

Sometimes, astronomers need to refer to the stars in the spiral arms in terms of a rectangular coordinate system as indicated respectively by the numbers in gray along the bottom and left side of the diagram. The square containing the rectangular coordinates will always be the smallest odd-numbered perfect square that will contain the star. For example, the star in spiral location `22` would be in a 5x5 square since 25 is the smallest odd-numbered perfect square greater than 22. Spiral star number `22` would have rectangular coordinates `4 5` as shown in the diagram above.

You have been asked to write a program that will print the rectangular coordinates of a star given its spiral location in a galaxy.

Input

The only line of input will contain a list of positive integers x ($1 < x \leq 625$), separated by a single space. Each integer in the list will be spiral location of a star.

Output

For each spiral location input, you will print the star's location using rectangular coordinates, one star per line, and the specifications above.

Example Input File

```
22 33 13
```

Example Output to Screen

```
4 5
3 1
1 1
```

12. Unique Prime Years

Program Name: Unique.java Input File: unique.dat

There are many "prime" years in the 21st century. A prime year is any year that is a prime number. There are some prime years that have other unique characteristics. For example, 2053 is a prime year which can be written as the sum of some number of consecutive prime numbers: 199 + 211 + 223 + 227 + 229 + 233 + 239 + 241 + 251 = 2053. Some prime years are even more unique because they can be written as the sum of two or more sets of consecutive prime numbers.

You are to write a program that will determine if a given a year in the 21st century is a prime year or a unique prime year with special characteristics as outlined above.

Input

The first line of input will contain a single integer `n` that indicates the number of years to follow. The following `n` lines will each contain a single year of the 21st century, 2000-2099.

Output

For each year of input, you will print the statement that best describes each of the years:

- *yyyy* `PRIME YEAR`
- *yyyy* `PRIME YEAR AND THE SUM OF` *x* `CONSECUTIVE PRIMES`
- *yyyy* `PRIME YEAR AND THE SUM OF MORE THAN ONE SET OF CONSECUTIVE PRIMES`
- *yyyy* `NOT A PRIME YEAR`

Note: In the output above, *yyyy* is the year given and *x* is the number of consecutive primes whose sum equals the year.

Example Input File

```
4
2053
2012
2087
2099
```

Example Output to Screen

```
2053 PRIME YEAR AND THE SUM OF 9 CONSECUTIVE PRIMES
2012 NOT A PRIME YEAR
2087 PRIME YEAR
2099 PRIME YEAR AND THE SUM OF MORE THAN ONE SET OF CONSECUTIVE PRIMES
```