ACM Programming Contest
Dixie State University
March 1, 2014

Do NOT turn this page until

10:00 am Mountain Standard Time!

Rules

1. Each team consists of up to three students.
2. Each team may use one provided computer.
3. Teams may use any printed references.
4. Teams may not use any electronic aids, including the internet. The help system built in to your programming environment is okay. Official documentation websites such as docs.python.org are okay.
5. High school teams and college teams compete and are judged separately (unless otherwise requested).
6. All solution must be written in C++ or Java. Python is also permitted for high school and middle school teams, and for college teams where no member has completed CS 3005 or any other course that requires C++.
7. The team with the most correct solutions wins.
8. Ties will be broken using a time score:
   (a) Each time a team submits a correct solution, the number of minutes that have elapsed since the beginning of the competition is added to the time score.
   (b) For each incorrect solution submitted, a 20 minute penalty is added to the time score, but only if the team eventually submits a correct solution to that problem.
   (c) Multiple penalties will be added for multiple incorrect solutions to the same problem.
9. The input for each problem comes from standard in. This goes by the names `cin` and `STDIN` in C++, `System.in` in Java, and `sys.stdin` in Python. Other methods may also exist for receiving input from standard in for each language.
10. The output for each problem should be sent to standard out. This goes by the names `cout` and `STDOUT` in C++, `System.out` in Java, and `sys.stdout` in Python. Other methods may also exist for sending output to standard out for each language.
11. The output of submitted solutions must exactly match the output of the reference solution, down to the last character. Whitespace differences matter. Any other output, including debugging output, may cause an otherwise correct solution to be marked as incorrect. Each problem statement with example input and output shows exactly where newline characters are placed and where spaces are appropriate.
12. Solutions have a 10 second time limit. Any solution that runs longer than that will be considered incorrect.
13. The contest begins at 10:00 am and ends at 3:00 pm.

Contents

1 SPECTRE Prime

The SPecial Executive for Counter-intelligence, Terrorism, Revenge and Extortion has a new numbers racket to raise money for their nefarious purposes. They publish a list of "seed" numbers and then allow people to bet on the "companion" number for each seed.

You notice a pattern in their system. The companion numbers are always the largest prime number less than the seed number. You help the 007 to bet big and win by creating an app for his jWatch device that will give the companion numbers for a list of seed numbers.

The input will have one seed number per line. There will be no more than 32000 lines in the input. Each seed number will be no more than 32000, and will be at least 3.

The output from your program will be one line per input line, with the seed number and companion number displayed, separated by 1 space.

A prime number is defined to be a number that is evenly divisible only by itself and 1.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
12↵
45↵
78↵
150↵
34↵
67↵
17↵
```

Sample Output

```
12 11↵
45 43↵
78 73↵
150 149↵
34 31↵
67 61↵
17 13↵
```

2 Self Destruct Codes

A villain's base self destruct system is locked by a simple numerical code. The villain is forgetful but thinks he is very clever. You have been able to deduce how the villain comes up with his codes, because of how much he likes to brag about it. You have also been able to determine the digits included in his code by worn number pads elsewhere in the base.

The code is a prime number from the fibbonacci sequence that is at most 16 digits. In order to obfuscate the code, the villain reseeds the fibonacci sequence's initial numbers.

The fibonacci sequence is a series of numbers where each new number is found by adding the previous two numbers before it.

You will be given the first two numbers of the villain's custom fibonacci sequence, followed by all the digits the number may contain.

Output all the possible codes seperated with a space. Output "No Code" if it is not possible.

Note: remember that 1 is not a prime number.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
0 1 2345↵
0 1 2480↵
3 9 27↵
5 7 0123456789↵
```

Sample Output

```
2 3 5 233↵
2↵
No Code↵
5 7 19 31 131 1453 2351 42187 1981891 3206767 13584083 332484016063 66165989928299↵
```

3 Sweetie Smasher

MI6 spends millions of pounds every year settling sexual harassment lawsuits filed against James Bond. In a desperate attempt to stem the tide, Q is writing a game for Bond's smartphone, hoping that he will keep his hands on his phone a bit more and on the ladies a bit less.

The game is called "Sweetie Smasher", and it is played with a grid of sweets on the screen. When the player lines up three or more sweets of the same color in a row or column, they are smashed and disappear. The remaining sweets fall down to fill in the holes, and new ones fall from the top.

You must help Q complete the game by writing the code to smash sweets. The input is a snapshot of the game in progress. Each letter represents the color of a sweet. You must identify and destroy any sequence of three (or more) duplicate characters in a row or column. You should find and destroy all such sequences at the same time. Any holes that result should then be filled in by gravity, i.e., the pieces above a hole should fall down. Drop in space characters from above to fill in the holes.

After the rows/columns of three or more sweets have been destroyed and gravity drops the others down, new rows/columns of three or more of the same color may form. Smash them and repeat until there are no further changes to the board. The output is the final board, including spaces.

Each game board will contain at most 100x100 sweets. Boards are rectangular, but not necessarily square.

Sample input and output are shown on the next page.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
ogrrgggo↵
ogbbgrbo↵
bbgrrggg↵
boooooorg↵
ogbrogbr↵
bbbrgorb↵
bboogrgb↵
ogogoggb↵
```

Sample Output

```
        ↵
o       ↵
ogr     ↵
bgb g b ↵
bbgrogbo↵
ogbbgoro↵
bboogrgg↵
ogogoggr↵
```

4 Obfuscinator

This problem had technical difficulties on the judging system. It was not actually used in the contest. Nevertheless, it is here for your coding pleasure.

Q has given James Bond his latest gadget... the obfuscinator! This simple gadget allows Bond to make it difficult for bad guys to understand what his messages say when he transmits them back to HQ. The obfuscinator works simply enough though, it reads each character and replaces that character with a different unicode character. The formula you should follow is this:

- convert each character to its decimal equivalent
- add 1264 to the character (this is our secret 'KEY')
- convert that decimal number to its unicode equivalent
- add the new unicode character to the output string
- should be done for all lines in the file
- encode everything except white space (space, tab, newline)

Many of you have perhaps heard of ASCII. This was the original character set that went through various changes. The idea of ASCII was that each character that you type could be represented in a binary value. The original creators only allocated 8 bits for the storage of an ASCII character. This resulted in only being able to represent 256 unique characters. This became a problem when people desired to represent non latin characters (i.e. Russian, Chinese, Japanese, and many other languages).

Unicode has several different variations as well. Essentially though it is a 16-bit character set which means you now have 65,536 distinct bit combinations that can represent characters. So, now we can represent the characters found in many langugages.

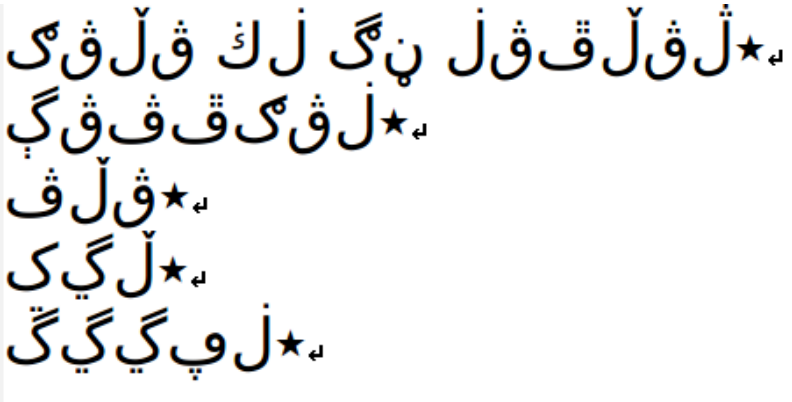
Sample input and output are on the next page.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
Here is my secret↵
peaches↵
are↵
for↵
noobs↵
```

Sample Output



Technical Notes

- For Python unicode characters should print well both in IDLE and at command line(Tested in Linux), refer to (http://docs.python.org/2/howto/unicode.html)
- For C++ in Linux refer to (http://stackoverflow.com/questions/18040393/printing-unicode-characters-c-linux)
- For C++ in Windows refer to (http://stackoverflow.com/questions/15911141/writing-unicode-to-a-file-in-c). Note that in Windows you may not be able to see the Unicode character at the console, but it should write to a file and you can view it in Notepad.

5 Diamonds Are Forever

Bond is tasked with investigating a major diamond smuggling ring which begins in Africa and runs through Holland and the UK to the United States. Disguised as professional smuggler and murderer Peter Franks, Bond travels to Amsterdam to meet Peter's contact.

In order to pull off the charade, Q must supply Bond with a diamond making machine that will allow him to make diamonds of any needed size from 1 carat up to 1000 carats. However, Q is in the middle of a project involving a vaporizing gun that can bounce off walls and turn corners, so he hires you as a subcontractor to write the Diamond Making Program.

Your program should work as follows: The input consists of one or more integers, each on its own line. For each integer, you are to create (print) a diamond that big. (The diamond sizes will always be between 1 and 1000 inclusive.)

On the next page is a sample input, along with the corresponding output your program should print:

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
3↵
2↵
1↵
8↵
```

Sample Output

```
  *↵
 * *↵
* * *↵
 * *↵
  *↵
 *↵
* *↵
 *↵
*↵
       *↵
      * *↵
     * * *↵
    * * * *↵
   * * * * *↵
  * * * * * *↵
 * * * * * * *↵
* * * * * * * *↵
 * * * * * * *↵
  * * * * * *↵
   * * * * *↵
    * * * *↵
     * * *↵
      * *↵
       *↵
```

Your spacing must be exactly correct in order to fool the bad guys. Have leading spaces on each row before the '*' symbols start. Note that the row representing the middle of the diamond will have no leading spaces. Have one space between each '*' symbol. After the last '*' on a row, there should be NO SPACES, just a return. There should not be any extra returns between multiple diamonds.

PYTHON Hints: To print one '*' followed by one space, use: print '*', To print one '*' followed by one return, use: print '*'

6 The Stakeout

Bond investigates the murder of 009, killed in East Berlin while dressed as a circus clown and carrying a fake Fabergé egg. An identical egg appears at auction and Bond establishes the buyer, exiled Afghan prince, Kamal Khan is working with Orlov, a renegade Soviet general, who is seeking to expand Soviet borders into Europe. Bond meets Octopussy, a wealthy woman who leads the Octopus cult. Bond finds out that Orlov has been supplying Khan with priceless Soviet treasures, replacing them with replicas, while Khan has been smuggling the real versions into the West, via Octopussy's circus troupe.

Bond and Felix are assigned to observe the Octopussy complex, noting everyone who enters and exits. At any given point, they may be asked to give an accounting of exactly who is currently on the premises. It turns out that despite their excellent marksmanship, and James Bond's amazing "people" skills, neither one can remember more than 3 names at a time, let alone the up to 100 residents that might be in the mansion at any given time. Therefore, you must write them an application they can use to track who enters and exits, and then reports, in alphabetical order, everyone who is currently on the property. Note that at the beginning of the stakeout, the only individual on the premises is Octopussy herself.

The input file consists of "Enter" and "Exit" lines, either of which is followed by a colon and a space, then one or more persons. Multiple people are separated by commas, except when there are only two. Given two or more one a line, the last is preceded by the word "and." All lines end with a period.

Sample input and output are shown on the next page.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
Enter: Khan and Orlov.↵
Exit: Octopussy and Orlov.↵
Enter: Jaws, Oddjob, Blofeld, and Moon.↵
Enter: Octopussy and Bond.↵
Exit: Bond.↵
Enter: Drax, Goldfinger, Big, Largo, No, Carver, Klebb, and Elektra.↵
Exit: Big, Carver, Zorin, and Octopussy.↵
Exit: Goldfinger and Khan.↵
Enter: Zorin, White, Stromberg, and Koskov.↵
Enter: Whitaker, Sanchez, Chang, Octopussy, and Khan.↵
Exit: Elektra, Blofeld, Moon, No, Oddjob, Zorin, White, and Stromberg.↵
```

Sample Output

```
Chang↵
Drax↵
Jaws↵
Khan↵
Klebb↵
Koskov↵
Largo↵
Octopussy↵
Sanchez↵
Whitaker↵
```

7 I Expect You To Die

You need to analyze a body of text to find the most repeated words, just so that you can pretend to know more than you really do in case you are set up to be executed by some complicated and inefficient device which gives you time to argue for your life.

You will be given a body of text as input. Find the most commonly used three word phrase. A phrase is three consecutive words in a sentence. For example "See Jane run fast." has 2 three word phrases, "See Jane run" and "Jane run fast". The input text will consist of up to 50,000 words and punctuation, with each words less than 100 characters in length.

Sentences may be ended by '.', '!', or '?'. Ignore all other punctuation, such as commas, semicolons or colons between words. Also, treat all letters as lower case.

If there is a tie, use the phrase with the most characters. If there is still a tie, use the phrase that comes alphabetically first.

The output is all in lower case, and contains the three word phrase a comma and the number of times the phrase occurs in the text.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
The James Bond series focuses on a fictional character created↵
in 1953 by writer Ian Fleming, who featured him in twelve novels↵
and two short-story collections. Since Fleming's death in 1964↵
seven other authors have written authorised Bond novels or↵
novelizations: Kingsley Amis, Christopher Wood, John Gardner,↵
Raymond Benson, Sebastian Faulks, Jeffery Deaver and William↵
Boyd. Additionally, Charlie Higson wrote a series on a young↵
James Bond, and Kate Westbrook wrote three novels based on the↵
diaries of a recurring series character, Moneypenny. Several↵
comic book adaptations of the James Bond films have been published↵
through the years.↵
```

Sample Output

```
the james bond, 2↵
```

8 Virus

An international villain's virus has begun infecting the population of a major city. Surprisingly, the virus acts very predictably in how it infects city blocks. So much so that you have been asked to provide a computer model to predict the spread of the infection over time. The city is a grid of blocks either infected or not. The pattern observed has been that:

- Any single block that is infected can be cured if less than two of the surrounding eight blocks are infected for that day.
- Any infected block with two or three infected blocks surrounding it remains infected despite any attempts to cure it in that day.
- Any uninfected block with exactly three surrounding infected blocks becomes infected.
- Any infected block with more than three surrounding infected blocks becomes uninfected because all its victims die.

Due to the premium of space in the city, any cell which is depopulated by the virus will repopulate, despite quarantine and the risk of infection.

Your program will be given an input of a number of days and map of the city and currently infected blocks. The input map will be of the format that '`-`' is uninfected and '`#`' is an infected block. The city input map is no more 1,024 blocks on each side and is rectangular. You will output the total number of infected cells and a map of the infection after the specified number of days. You must assume that the city is effectively infinite in every direction for the purposes of calculating neighbor relationships and total count, however you will only render a map of the original input area for your output.

Several sample inputs and outputs follow:

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input 1

```
3↵
-#-↵
-#-↵
-#-↵
```

Sample Output 1

```
3↵
---↵
###↵
---↵
```

Sample Input 2

```
2↵
-----#↵
--#-#-↵
---#--↵
```

Sample Output 2

```
4↵
------↵
---##-↵
---##-↵
```

Sample Input 3

```
1↵
--#↵
--#↵
--#↵
```

Sample Output 3

```
3↵
---↵
-##↵
---↵
```

9 MI6 Field Encryption

During training with MI6, all agents are required to demonstrate high proficiency in math– square roots in particular. The true reason is kept a secret; however, it is the key to a method used for encrypting and decrypting classified mission data and communications with their field agents.

To encrypt the message they process one line at time like this:

1. Write out the line.
2. Convert everything to lowercase.
3. Remove all spaces.
4. Find the smallest integer, r, such that $r^2 \geqslant s$, where s is the size of the line without spaces.
5. Write the line down in columns of r characters from left to right.
6. Fill out the square with arbitrary letters.
7. Create the encrypted line by adding each row. from the square of letters, the first row left-to-right, the next row right-to-left, and so forth.

For example, if the line is "This is my super secret data!" They would write this square:

```
t s p c a
h m e r t
i y r e a
s s s t !
i u e d s
```

Then add the rows to produce this line:

```
tspcatremhiyrea!tsssiueds
```

MI6 recently received a very lengthy message from an agent they believed to be MIA. Your job is to recover for MI6 the original message (along with any extra padding characters).

Input

The first line of the input is an integer (k) telling you how many lines the message contains. There will be several ($1 \leqslant k \leqslant 1500$) lines in the message. Process each line separately.

Output

Each line of the original message should generate one line of output, giving the original plaintext message (lower case with no spaces) and any additional letters added for padding the message.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
3↵
tspcatremhiyrea!tsssiuedc↵
shhtvejeapivborbbeeestsep↵
css,dosceineitcomihtcmeadrurmeepunbayc.erihpkecteesetemiieaistir↵
```

Sample Output

```
thisismysupersecretdata!c↵
spieshavethebestjobeverbp↵
computerscienceisthebest,itmakesdecryptionmucheasier.imeceaderii↵
```

10 Secret message

Agent 007 played golf with a supervillain's henchman, and managed to steal a coded message from him. After studying the message, 007 worked out the code. The message was there in plain text, but it was obscured by a bunch of junk characters. By removing the junk characters, the secret message could be read clearly.

Help 007 by writing a program to remove the junk characters from a message. Leave all whitespace alone, but find the 10 most commonly-occuring characters in the message and remove them. The output is the original message with the junk removed. You may assume that all data is from the ASCII character set.

An input file will have fewer than 1,000 lines. Each input line will have less than 20,000 characters.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
]W]]W]""WJ,J3"33J,Bk3zJ,]23z],WW,]/WW]zJ3z]Wzi"BlW,WWJ2"l/↵
J"2]]2J","2J]0""J,3/230Wz2"7BW2zJ"",],2J""2W//3,W,"23]]z/J"z223WWJ,"2"J↵
```

Sample Output

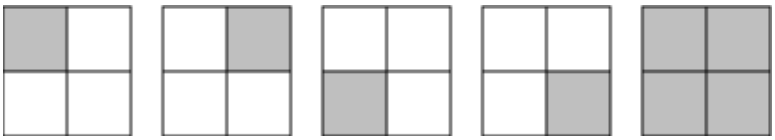
```
kill↵
007↵
```

11 Fun and Critical Squares

While on a mission you made contact with the leader of an international crime group who you suspect are planning a missile strike. Earlier in the week the agency recovered a message sent from the leader to a militant group in North Korea containing several numbers that appear to be some form of encryption. From your intell, you know that the life-long addiction of the leader was solving and making puzzles, locks, and cyphers.

While at the bar you sit next to the leader and notice doodling on his napkin. As you leave you grab it without being noticed. When you get back to you room and pull out the napkin you read, "how many different squares are there in a grid of $N \times N$ squares?".

On the same napkin there was a drawing which is reproduced below, showing that, for $N = 2$, the answer is 5.



You suspect he is using this puzzle as a cypher for the launch and abort codes so you write a program so you can use it on your next mission.

The input contains several test cases. Each test case is composed of a single line, containing only one integer N, representing the number of squares in each side of the grid ($1 \leqslant N \leqslant 200$). The end of input is indicated by $N = 0$.

For each test case in the input, your program must print a single line, containing the number of different squares for the corresponding input.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
1↵
2↵
4↵
20↵
200↵
0↵
```

Sample Output

```
1↵
5↵
30↵
2870↵
2686700↵
```

12 Moonraker

An evil scientist has invented a nerve gas that will kill people at an alarming rate. It kills 1 person in the first minute, $1 + 2$ people in the second minute, $1 + 2 + 3$ in the third minute, and so forth.

You need to look smart on the screen, so you need to be able to quote how many people will die in the N^{th} minute, where $1 < N \leqslant 10,000$. Of course, you write a program to help you out, just don't let your cheat sheet show on screen.

The input to your program contains one number (the number of minutes) per line. There will be at most $10,000$ lines in the input.

The output from your program will have one line per input line. The output line will contain the input number (number of minutes) followed by the number of people who will die that minute, separated by a single space.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
10↵
100↵
1000↵
35↵
50↵
3456↵
123↵
1↵
5↵
37↵
```

Sample Output

```
10 55↵
100 5050↵
1000 500500↵
35 630↵
50 1275↵
3456 5973696↵
123 7626↵
1 1↵
5 15↵
37 703↵
```