ESSS Coding Problems

General Instructions

Please read the instructions below carefully. Besides reading the submitted code, we also run automatic tests on it and the tests make certain assumptions.

- All **four** coding problems must be solved.
- Languages that can be used to implement the coding problems are:
  - ❏ Python 3
  - ❏ C++
  - ❏ Java

  You can choose a different language for each problem if you like.

- **Non-interactive** command-line programs, with all inputs being given as command-line parameters (“argv”) and outputs printed in the terminal. *Do not* make the programs interact with standard input.
- Only the **standard library** of the language(s) of choice can be used, no external libraries.
- Programs should return **0 (zero) exit code** when it executes normally and without errors.
- Programs that return **0 (zero) exit code** must **output exactly what asked and that output only**. We use automated test to check the examples. For example, if the problem states *“print the number in a single line”*, then:

  Do this:

```
$ ./main 10
1
```

  Do not do this:

```
$ ./main 10
The result is = 1
```

- **Errors must be adequately treated**: the exit code of the program **has to be != 0** if it identifies any error, and a helpful message should be printed (the actual message doesn’t matter, only the exit code being non-zero is mandatory). Note that specific error treatment is dependent of the problem, so read each description carefully.
- Add documentation and comments as you see fit.
- Use plain ASCII encoding (or UTF-8 if Unicode is needed).
- Submit your code in **zip** or **tar.gz** formats by e-mail.
- **Layout**: the code must be organized in a folder structure **exactly** as follows, and each problem must have at least one file. The file should be called “Main.java” for Java, “main.cpp” for C++ and “main.py” for Python.

```
ESSS
    ESSS Coding Problems.pdf
    ESSS1
        main.cpp
```

- ESSS2
    - main.cpp
- ESSS3
    - main.cpp
    - sample1.txt
    - sample2.txt
- ESSS4
    - main.cpp

- Do not declare a package when using Java (it will be run using `java Main <arguments>`).
- C++ source can be up to **C++11 compatible** and should only use standard C++, do not use compiler extensions or platform-specific libraries.
- **Include this document** at the root ESSS folder, as shown above.

ESSS1 (Fizz Buzz)

Write a program that prints the numbers from X to Y. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.

Input

Range: X and Y are positive integers that must be given by the user, between 1 and 2000; Y must be greater than X.

Output

Each number evaluation must be printed in a single line.

Sample (correct input, success)

```
$ ./main 1 15
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz

$ echo $?
0
```

Sample (incorrect input, failure)

```
$ ./main 1 4000
Invalid input, Y > 2000

$ echo $?
1
```

ESSS2 (A List Game)

You are playing the following simple game with a friend:

1. The first player picks a positive integer X.
2. The second player gives a list of k positive integers $Y_1,\dots,Y_k$ such that $(Y_1+1)(Y_2+1)\cdots(Y_k+1)=X$, and gets k points.

Write a program that plays the second player.

Input

The input consists of a single integer X satisfying $10^3 \le X \le 10^9$, giving the number picked by the first player.

Output

Write a single integer k, giving the number of points obtained by the second player, assuming she plays as good as possible.

Sample (correct input, success: exit 0)

```
$ java main 65536
16

$ java main 127381
3

$ java main 725594112
23

$ echo $?
0
```

Sample (incorrect input, failure)

```
$ java main 15
Value must be between 10^3 and 10^9

$ echo $?
1
```

ESSS3 (Charting Progress)

You’re consulting for a fitness training institute which is working on becoming more technologically adept. Currently, it has an old computer which tracks various statistics about athletes in training. Each day, the athlete logs in and records a single number, which is information about their workout. For example, a runner may record the number of miles run. The computer simply stores this until the trainer requests a log for an athlete. The computer then prints a log in text format indicating each record.

The original log is in column-major format, where each column represents one record (such as miles run that day). Since the records may fluctuate, the trainer wants a log that is easier to analyze. Your job is to write a program which orders the records (columns) from least to greatest value (asterisks progressing from the lowest to the highest rows), so that the trainer can get a different, smoother view of an athlete’s progress.

Input

Input is a series of at most **100** logs that should be processed. Each log is a rectangular block of text of size at most **80** columns and **100** rows (one row is one line). Most of the characters are periods (.), but each column has *exactly one* asterisk (*) indicating the value recorded for the day. Each column in a log has the same height.

Between each pair of logs **is a single blank line**. Input ends at the end of the file.

The log file must be passed as the only **argument** to the program.

Output

For each log, print out the log in the desired (sorted) order. Print a blank line between each pair of logs printed.

Sample (correct input, success)

sample1.txt

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

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

```
$ python main.py sample1.txt
...................*
.................**.
..............***...
........******......
......**............
.....*..............
..***...............
....................
**..................

..........
......****
..****....
..........
**........
$ echo $?
0
```

Sample (incorrect input, failure)

sample2.txt

```
...a
**..
..**
```

```
$ python main.py sample2.txt
Invalid character found: ‘a’

$ echo $?
1
```

ESSS4 (Square Root)

In this question we will implement one simple way to calculate the square root of a positive number, read the instructions carefully and **do not** use functions from libraries (such as sqrt() in C/C++).

A simple way to calculate the square root of a positive number *X* consist of:

1. Choose two numbers, *S* and *E*, in a way that $\sqrt{X}$ belongs to the interval [*S*, *E*].
2. Choose a precision desired ε (small precision will result in a better approximation).
3. How we must have that $\sqrt{X} \in [S, E]$, one approximation to $\sqrt{X}$ is the middle value of the interval, *M*.
4. While the length of our current interval [*S*, *E*] is greater than ε, we will repeat the process:
    a. Find the middle value of the current interval, *M*.
    b. If $M^2 > X$, it means that $\sqrt{X} \in [S, M]$ so we change the value of *E* to *M*.
    c. Otherwise we have that $\sqrt{X} \in [M, E]$ so we change the value of *S* to *M*.
5. When the length of our interval becomes smaller than ε, the middle value *M* will be a good approximation of $\sqrt{X}$.

Write a program to calculate the square root of an integer number, using the method described above and a precision of **10^{-5}**.

Input

An integer number greater than 0.

Output

The square root value with **3 decimal** places, including trailing zeros if any.

Sample (correct input, success)

```
$ python main.py 10
3.162

$ python main.py 20
4.472

$ python main.py 9
3.000

$ echo $?
0
```

Sample (incorrect input, failure)

```
$ python main.py not_a_number
Input must be a number

$ echo $?
1

$ python main.py
Missing number argument

$ echo $?
1
```