FINDING SUBSTRINGS

In this exercise, you must write a program that receives two strings. Your program must then output 'YES' if a common substring exists or 'NO' if there is no common substring exists.

Your program should prompt for each string at launch. The input should be as follows, input the first string then press the enter key. Input the second string then press the enter key. Each string may be up to 10000 characters. You can safely assume that a substring can be as simple as a single character. You can also assume that each string will only include ASCII characters.

Your program's output must match the examples given below exactly. Your program must not prompt for input. Note carefully the spelling, capitalization, punctuation, and spacing of the output. The input that will be given to your program is highlighted in red.

Example Run 1

"Hello World"
"Bye World"
YES

Example Run 2

"abcdefg"
"hijklmn"
NO

CORRECTIONS

For this exercise, you must write a program that accepts a string of input followed by a correction that needs to be made to the input. Your program must then output the corrected string.

Your program will receive input as soon as it launches. The first input will be a phrase, sentence, or paragraph, followed by the enter key. The second input will be the subset of the first input which needs to be replaced. The third input will be the new text that should replace the text given in the second input. Your program must then output the corrected version of the first input, with all instances of the second input replaced by the third input.

You may safely assume that the phrase, sentence, or paragraph your program will be given will be no more than 10,000 characters and will not contain any line breaks. The input given to your program will be encoded as UTF-8 and may include non-ASCII characters (e.g. á, è, ü, “, ”, etc.).

*Your program's output must match the examples given below **exactly**. Your program must not prompt for input. Note carefully the spelling, capitalization, punctuation, and spacing of the output. The input that will be given to your program is highlighted in **red**.*

EXAMPLE RUN 1

```
“To be or not to be, that is the potato.”
potato
question
“To be or not to be, that is the question.”
```

EXAMPLE RUN 2

```
Old King Cole was a dairy cold sole, and a dairy cold sole was he.
dairy cold sole
merry old soul
Old King Cole was a merry old soul, and a merry old soul was he.
```

VOLUME OF A SPHERE

For this exercise, you must write a program that determines the missing value from the calculation of the volume of a sphere.

Given the radius (r) of a sphere, the volume (V) is calculated using the following formula:

$$V = \frac{4}{3}\pi r^3 \tag{1}$$

Therefore, if given the volume (V) of a sphere, the radius (r) can be calculated with the following formula:

$$r = \sqrt[3]{\frac{3V}{4\pi}} \tag{2}$$

Note also that the cube root in equation 2 can be expressed as an inverse exponent:

$$r = \sqrt[3]{\frac{3V}{4\pi}} = \left(\frac{3V}{4\pi}\right)^{\frac{1}{3}} \tag{3}$$

Your program will receive input as soon as it launches. The first input will be either the word "volume" or the word "radius" to indicate which variable will be given, followed by the enter key. The second input will be a number indicated the value of the variable. Your program must then output the name of the value that was calculated, an equals sign (=), and the calculated value rounded to the nearest tenth (i.e. one digit after the decimal).

You may safely assume that the first line of input will consist only of either "volume" or "radius". You may safely assume that the second line of input will be a non-negative integer or floating point number less than 10,000 and that it will be given without thousands separators (such as a comma) and without units. You must use 3.14159 as the value of π.

*Your program's output must match the examples given below **exactly**. Your program must not prompt for input. Note carefully the spelling, capitalization, punctuation, and spacing of the output. The input that will be given to your program is highlighted in **red**.*

EXAMPLE RUN 1

```
radius
5
volume = 523.6
```

EXAMPLE RUN 2

```
volume
523.6
radius = 5.0
```

VOTES

For this exercise, you must write a program that tabulates the results of officer elections in your chapter.

Your program will receive input as soon as it launches. The first input will be the number of ballots to process, which we will call *n*. Your program will then receive *n* ballots, each on a separate line of input. Each ballot will consist of 6 names separated by a space. The names represent that ballot's choice for each of the six standard officer positions. The positions will always be in the following order: President, Vice President, Secretary, Treasurer, Reporter, and Sergeant-at-Arms. Your program must then output the name of the winner for each office (in the same order just noted), followed by the number of votes. If there is a tie, your program must output "-tie-" followed by the names of the candidates who tied for that position in alphabetical order, followed by the number of votes. In all cases, the number of votes must be separated by a comma.

You may safely assume that the number of ballots will be between 1 and 1,000 (inclusive). Each ballot will be no more than 10,000 characters and will not contain any line breaks. The input given to your program will be encoded as UTF-8 and may include non-ASCII characters (e.g. á, è, ü, “, ”, etc.).

*Your program's output must match the examples given below **exactly**. Your program must not prompt for input. Note carefully the spelling, capitalization, punctuation, and spacing of the output. The input that will be given to your program is highlighted in **red**.*

EXAMPLE RUN 1

```
4
Camille Irene Stan Mitch Katrina Nate
Camille Diane Stan Keith Katrina Nate
Camille Irene Stan Keith Katrina Nate
Camille Irene Stan Mitch Katrina Nate
President: Camille, 4
Vice President: Irene, 3
Secretary: Stan, 4
Treasurer: -tie- Keith Mitch, 2
Reporter: Katrina, 4
Sergeant-at-Arms: Nate, 4
```

EXAMPLE RUN 2

```
7
Camille Irene Stan Mitch Katrina Donna
Camille Diane Stan Keith Katrina Nate
Camille Irene Stan Keith Katrina Donna
Camille Irene Stan Mitch Katrina Nate
Flora Diane Stan Keith Katrina Nate
Flora Irene Stan Keith Katrina Donna
Camille Irene Stan Mitch Katrina Donna
President: Camille, 5
Vice President: Irene, 5
Secretary: Stan, 7
Treasurer: Keith, 4
Reporter: Katrina, 7
Sergeant-at-Arms: Donna, 4
```

ROBOT MOVES

For this exercise, you must write a program that identifies the position of a robot after it makes a programmed series of moves.

The robot will be moving in an area that has been laid out in a grid. You will be given the number of columns and rows in the grid. The columns will be labeled with capital letters, starting with "A" and continuing in alphabetical order. The rows will be labeled with numbers, starting with 1. You will then be given the starting position of the robot as the column followed by the row. For example, suppose you are given an area with 6 columns and 6 rows, with a starting position of "D3". The resulting area would look something like this:

	A	B	C	D	E	F
1						
2						
3				☺		
4						
5						
6						

Your program will receive input as soon as it launches. The first input will be the number of columns. The second input will be the number of rows. The third input will be the starting position of the robot, as a grid reference (i.e. a column letter followed by a row number). The fourth input will be a sequence of letters, indicating the movements of the robot. Each letter will be one of "N", "E", "S", "W", indicating one of the four cardinal directions, north (meaning towards the top of the grid), east (toward the right of the grid), south (toward the bottom of the grid), and west (toward the left of the grid). Each letter indicates a move of exactly one space. Your program must then output the final position of the robot as a grid reference. If the robot has at any point moved beyond the confines of the grid, your program must output the phrase, "Out of bounds."

You may safely assume that the number of columns, rows, and moves will not exceed 1,000 each. You may also safely assume that the given starting position will be a valid grid reference.

*Your program's output must match the examples given below **exactly**. Your program must not prompt for input. Note carefully the spelling, capitalization, punctuation, and spacing of the output. The input that will be given to your program is highlighted in **red**.*

EXAMPLE RUN 1

```
4
4
D4
NNNWWW
A1
```

EXAMPLE RUN 2

```
4
4
D4
NNNNWWWW
Out of bounds.
```