Contents

1 Checkerboard Flag

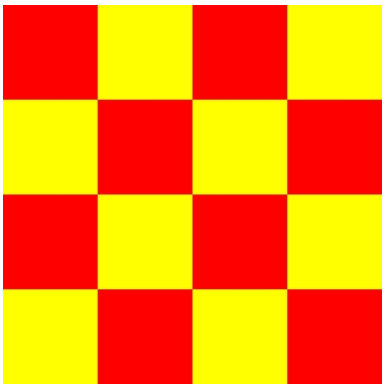
After raising his Orcish army, Saruman decides he needs a new banner to inspire his troops. After casting many enchantments, Saruman decides that a simple 2-color checkerboard pattern will bring him the most luck. Your job is to help Saruman design his flag.

You begin with some object-oriented code for drawing pictures. A "picture" is represented as a two-dimensional grid of colors, each of which contains a red, green and blue component in the range [0 - 255]. In these problems, a picture is defined by any class that implements the supplied Picture interface. The Picture interface's getWidth() and getHeight() methods defines the dimensions the picture, respectively. The getColor(int col, int row) method must specify the color of the picture at the specified position where (0, 0) specifies the top-left corner of the picture and the column number increases to the right and the row number increases downward.

Each color is defined by an instance of the supplied PictureColor class. A static utility method, PictureUtil.show() will display an instance of Picture on the screen. A supplied Image utility class can also be used for creating a Picture given a jpeg image file.

Complete the code for class CheckerBoard so it can generate a picture that is a checkerboard pattern of the two specified colors of the overall specified "width" and "height" dimensions. The checkerboard should contain (n x n) cells where n is the specified "gridSize" parameter. Cells in positions where the sum of the column number and row number are even should be "color1" and should otherwise be "color2". You are guaranteed that the dimensions of the checkerboard will be an integral multiple of "gridSize". A CheckerBoard object created with the following parameters looks like:

new CheckerBoard(256, 256, 4, PictureColor.RED, PictureColor.YELLOW);



Input Format

The input will be two integers. The first will be the height and width of the checkerboard. The second number will the number of cells in each dimension.

Output Format

The program should produce a checkerboard pattern corresponding to the input parameters.

Example

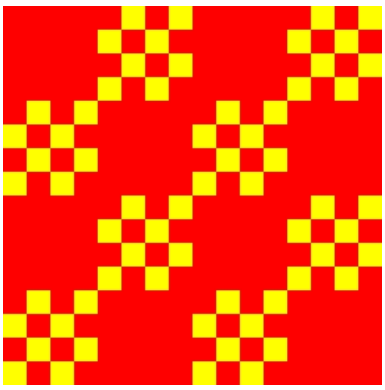
The input "256 4" produces the checkerboard picture shown above.

2 Complex Checkerboard

Saruman is pleased after seeing the results of your work in designing a checkerboard flag. However, after additional enchantments, Saurman decides that he would like to embellish the basic checkerboard flag further. Your task is design a new, more complex checkerboard flag.

You begin with some object-oriented code for drawing pictures. Complete the code for class ComplexCheckerBoard so it generates a picture that is a modification of the basic checkerboard pattern. The ComplexCheckerBoard should be the same as CheckerBoard except that "odd" cells (those that were drawn with "color2" should now be drawn with a new checkerboard (with the same number of cells) shrunk to fit in that cell. You are guaranteed that the dimensions of the checkerboard will be an integral multiple of the square of "gridSize". A ComplexCheckerBoard object created with the following parameters look like:

new ComplexCheckerBoard(256, 256, 4, PictureColor.RED, PictureColor.YELLOW);



Input Format

The input will be two integers. The first will be the height and width of the checkerboard. The second number will the number of cells in each dimension.

Output Format

The program should produce a complex 2-level checkerboard pattern corresponding to the input parameters.

Example

The input "256 4" produces the checkerboard picture shown above.

3 Alternating Letters

On the long long road to Mordor, Sam and Frodo get to talking ...

Sam: "Mr. Frodo, I am quite exasperated with our situation."

Frodo: "Sam, that is an interesting choice of words. Note that in the word 'exasperated' the letters 'e' and 'a' alternate 5 times."

```
e  x  a  s  p  e  r  a  t  e  d
↑     ↑        ↑     ↑     ↑
```

Sam: "Dear Mr. Frodo. We are starving and being hunted by nasty orcs and giant spiders, I find your sudden interest in spelling most counterintuitive."

Frodo: "My goodness Sam! You just topped yourself. The word 'counterintuitive' has fully 6 alternations, involving the letters 't' and 'i'!"

```
c  o  u  n  t  e  r  i  n  t  u  i  t  i  v  e
            ↑        ↑     ↑     ↑  ↑  ↑
```

Sam: "Master Frodo, we are starving. I want to go home! Please stop talking about such insignificant matters, or I will surely turn around and quit this instant."

Frodo: "Oh my goodness Sam, you just did it again. The word 'insignificant' also has 6 alternations!"

```
i  n  s  i  g  n  i  f  i  c  a  n  t
↑  ↑     ↑     ↑        ↑        ↑
```

Sam: "That does it. I'm heading back to the shire. You can keep the blasted ring."

Smeagol: "Sssilly hobbitssess."

Frodo loves to find alternating patterns of letters in words. An *alternation* is defined to be two distinct characters (letters, digits, or punctuation) that alternate at least three times as the word is read from left to right, as shown in the above examples. Thus, the word "boo" has no alternations, but "booboo" has "bobo" as an alternation of length 4. Upper and lower case characters are considered to be different, and so, "aragorn" contains the 4-element alternation "arar", but "Aragorn" only has the 3-element alternation "rar". Digits and punctuation may be used, so "a2b-c2d-" has the alternation "2-2-".

The objective of this problem is to write a program to find the length of the longest alternation in a word.

Input Format

Your program will read in a series of strings, one per line. Each string will contain no embedded spaces, but may contain punctuation symbols and digits.

Output Format

Your program will output the word and the length of the longest alternation. (It does not need to output the actual letters.) If there is no alternation of length 3 or more, then it outputs 0 as the length. See the example below.

Example

Input:	Output:
`exasperated`	`exasperated has 5 alternations`
`counterintuitive`	`counterintuitive has 6 alternations`
`insignificant`	`insignificant has 6 alternations`
`Aragorn`	`Aragorn has 3 alternations`
`a2b-c2d-`	`a2b-c2d- has 4 alternations`
`cat`	`cat has 0 alternations`
`ca`	`ca has 0 alternations`
`c`	`c has 0 alternations`

4 Maze Solver

Frodo and Sam are lost, wandering around the Fens outside of Mordor. All of the rocks and crags give the feeling of a maze. Having a guide would be helpful, but even more helpful would be methodical way to get out of the maze, to "solve it."

Your task is to help them do this. You will be given a maze where the entry to the maze is at the top, and the exit is at the bottom. Your goal is to write a function to find a path through the maze. To do this, you must implement a class `Maze`, having the following signature:

```
public class Maze {
  public final static int LEFT = 0;
  public final static int RIGHT = 1;
  public final static int UP = 2;
  public final static int DOWN = 3;

  public Maze(int width, int height, int startX, int endX);

  public void removeWall(int x, int y, int d);
  public void addWall(int x, int y, int d);
  public boolean hasWall(int x, int y, int d);
  public void setStartX(int x);
  public void setEndX(int x);
  public int getStartX();
  public int getEndX();
  public int getWidth();
  public int getHeight();

  public java.util.LinkedList solve();
}
```

You are free to use whatever datastructures you wish, but you must implement the methods exactly as listed above. The constructor creates a maze with the following characteristics:

- It has size `width` $\times$ `height`; The coordinate system in the maze places $(0, 0)$ in the upper left, and (`width` $- 1$, `height` $- 1$) in the lower right.
- The maze entry point is $(0,$ `startX`$)$ and whose exit point is (`height` $- 1$, `endX`).
- Every cell in the maze is surrounded by walls, such that all queries to `Maze.hasWall(...)` will return `true`.

The methods `hasWall`, `addWall`, `removeWall` all have a `direction` parameter; this should be one of `Maze.LEFT`, `Maze.RIGHT`, `Maze.UP`, or `Maze.DOWN`. The call `hasWall`(x, y, d) returns `true` if there is a wall in direction d at coordinates (x, y). The call `removeWall`(x, y, d) removes the wall in direction d (if any) at coordinate (x, y), while `addWall` adds one. The remaining methods should be self-explanatory. Note that you don't need to bother with error checking (though it might be a good idea to avoid bug hunting); you can assume that all x, y, d passed in are legal (i.e. within bounds of the maze, and referring to a valid direction).

Finally, the key method that you must implement is `solve`. This will return a path through the maze as a linked list of directions, starting from $(0, \texttt{startX})$ until it finally reaches $(\texttt{height} - 1, \texttt{endX})$. For example, given the following maze:

```
   0 1 2 3
  ***** ***
0 *   *S  *
  * ***** *
1 *  E    *
  *** *****
```

This is a 4×2 maze, with the starting point at S and the ending point at E. The correct answer would be *right* (to $(3, 0)$), *down* (to $(3, 1)$), *left* (to $(2, 1)$), and *left* (to $(1, 1)$, the end point). Textually, this path has the representation

```
>v<<
```

where `>` stands for *right*, `v` stands for *down*, `^` stands for *up*, and `<` stands for *left*. For your convenience, we have provided you with the class `MazeIO`:

```
public class MazeIO {
  public static Maze readMaze(BufferedReader in)
    throws MazeFormatException, IOException;
  public static void printMaze(Maze maze, PrintWriter out);
  public static void printPath(LinkedList path, PrintWriter out);
}
```

You can use these routines to read in a maze from standard input (`System.in`); it will call your implementation of `Maze` to create the maze as indicated. The `main` method you should use is given in `Solve.java`, also provided. Run your implementation by doing

```
java Solve < file
```

where `file` is the input file to use. The output will be maze followed by the path generated by your `solve` method. Output will be printed to `System.out`.

Examples

Input:[1]

```
4 2
***** ***
*   *S  *
* ***** *
*  E    *
*** *****
```

Output:

```
4 2
***** ***
*   *S  *
* ***** *
*  E    *
*** *****
>v<<
```

Input:

```
4 4
***** ***
* *  S* *
* * *** *
* *     *
* ***** *
* *     *
* * *****
*  E    *
*** *****
```

Output:

```
4 4
***** ***
* *  S* *
* * *** *
* *     *
* ***** *
* *     *
* * *****
*  E    *
*** *****
<v>>v<<v
```

Note that there may be multiple solutions for the same maze. Any correct solution will do.

[1] The first two numbers in the input file are the dimensions of the maze. You don't need to worry about this, since `MazeIO` takes care of formatting.

5 Maze Maker

Sauron enjoys making our heroes despair. One way to make them despair is to trap them in a maze, like the Mines of Moria. This leaves one problem: a methodical means for coming up with a maze.

Your task is to implement a maze-creation tool for Sauron. To do this, you will extend the `Maze` class above to include a new constructor:

```
public Maze(int width, int height, int startX, int endX,
            NumberGenerator gen);
```

This creates a maze using the following algorithm. Starting at (`startX`, 0), do a random depth-first search of the maze, knocking down walls as you search out the maze.

Here's how you do the depth-first search. Given a cell (x, y), consider all adjacent cells (either to the left, right, above, or below (x, y); no wraparound) that you haven't yet visited, and pick a random one (see below). Knock down the wall between the current cell and the chosen one, and then process the chosen cell. If there are no legal cells adjacent to (x, y), then you must backtrack to its parent (the one visited just before the current (x, y)), and repeat. When there are no more cells left to visit, you are finished. Finally, you should knock out the wall above the starting position, and the wall below the ending position.

Here's how you should randomly choose between cells. Of all the cells to consider, order them as follows: (left, right, up, down). For example, say that `startX` is 5, and you just starting the algorithm. Then you would consider cells $(4, 0)$, $(6, 0)$, and $(5, 1)$, in that order. Pick a random number between 0 and 2 to choose which. Now say you get random number 1, so that you pick $(6, 0)$, and you then start processing that. Now you can consider the cells $(7, 0)$ and $(6, 1)$ (you cannot consider $(5, 0)$ because you just came from there). And so on.

So that your output is reproducible, you will use an implementation of the interface `NumberGenerator` to calculate random numbers for the choices above:

```
public interface NumberGenerator {
    int genInt(int max);
}
```

That is, when you are choosing between cells, call `genInt` to get a number between 0 and `max`, inclusive. We will provide a sample implementation of this interface, `RandomGenerator`, that uses `java.util.Random` to generate random numbers.

Use the `main` method in the class `Construct.java` to test your algorithm. In particular, you should call it as follows:

```
echo width height startX endX seed | java Construct
```

All arguments are passed in through standard input (shown above using `echo`). Here, `width` is the width of the maze, `height` is the height, and `seed` determines the sequence of random integers to generate. The starting point in the maze will be `(startX,0)` and the ending point is `(endX,height-1)`. Both `startX` and `endX` should be between 0 and `width`-1. After running `Construct`, it will print out the generated maze, using the routine in `MazeIO`.

Examples

Input:

```
4 4 2 3 12001
```

Output:

```
4 4
***** ***
*   *S  *
* * *** *
* *   * *
* *** * *
*   * * *
* * *** *
* *    E*
******* *
```

Input:

```
6 6 4 5 1001
```

Output:

```
6 6
********* ***
*     *  S* *
* *** * *** *
*   *   *   *
*** ***** * *
* *   *   * *
* *** * *** *
*   * * *   *
* * * * * ***
* *   * *   *
* ***** *** *
*       *  E*
*********** *
```

6 Picture Unscramble

After successfully designing flags for Saruman, you are promoted and transferred to the signal corps for the Orcish army. A major responsibility of your new position is to find ways of passing information between troops in secret, without allowing pesky humans and elves to intercept your message. One of the tasks you are given is to encode images so that their contents may not be intercepted. Your predecessor already developed an algorithm for scrambling pictures, but was unfortunately injured by a hobbit wandering near the tower. Your job is to finished the task by developing an algorithm to unscramble images produced by the old scramble algorithm.

You must unscramble the specified input picture. The input picture is scrambled as follows. It is broken up into a grid of (numCols x numRows) blocks of pixels, and the blocks are randomly reordered by the supplied Scramble class. Your job is to place the blocks back in their correct position. The following two pictures show an input scrambled picture (with 10x10 blocks), and the resultant unscrambled result.


The input picture is guaranteed to be of dimensions that are an integral multiple of the number blocks along each axis. That is, the width of the picture is an integral multiple of the number of columns of blocks. The height of the picture is an integral multiple of the number of rows of blocks.

The correct position of each block is encoded within the top-left pixel of each block as follows. The red component of the pixel contains the correct col of that block, and the green component of the pixel contains the correct row of that block (with row and column starting at 0). So, for example, if the (red, green, blue) color of the top-left pixel of a block was equal to (1, 7, 73), then this means that this block should go at position (column=1, row=7).

For this problem, you must complete the supplied Unscramble class, generating an unscrambled picture, given an input scrambled picture as defined above with the specified number of rows and columns of blocks.

You may test your solution with the supplied UnscrambleTest class and the supplied "gollum.jpg" input image. Note that the complete source code for Scramble is supplied, but it is not necessary to look at that code in order to write Unscramble (although you are welcome to look at it).

Input Format

A string with the name of the input file.

Output Format

The original and unscrambled pictures.

7 Graphical Sequences

After wandering for days trying to escape a maze, you decide there must be better ways to analyze the properties of a maze. One suggestion was to consider the degrees of nodes in an undirected graph. Consider an *undirected graph* $G = (V, E)$ consisting of a set V of *vertices* and a set E of *edges*. We will consider only *simple* graphs, which are graphs in which there are no loops (edges of the form (x, x)) and in which there is at most one edge connecting any pair of nodes (i.e., $E \subset V \times V$). The *degree* $d(n)$ of a node $n \in V$ is the number of edges incident on it. For example, the degree of node 3 in Figure 1 is 2, while that of node 4 is 1. The sequence of numbers formed by listing the degrees of all nodes in a graph in non-increasing order is called its *degree sequence*. The degree sequence of the graph in Figure 1 is $S_1 = 3, 2, 2, 2, 1$. A sequence of nonnegative integers is called *graphical* if, when sorted in non-increasing order, it equals the degree sequence of *some* graph. Since the graph in Figure 1 has degree sequence S_1, it follows that S_1 is graphical, as are non-sorted versions of S_1, such as $S_1' = 2, 1, 3, 2, 2$ and $S_1'' = 2, 2, 3, 2, 1$.

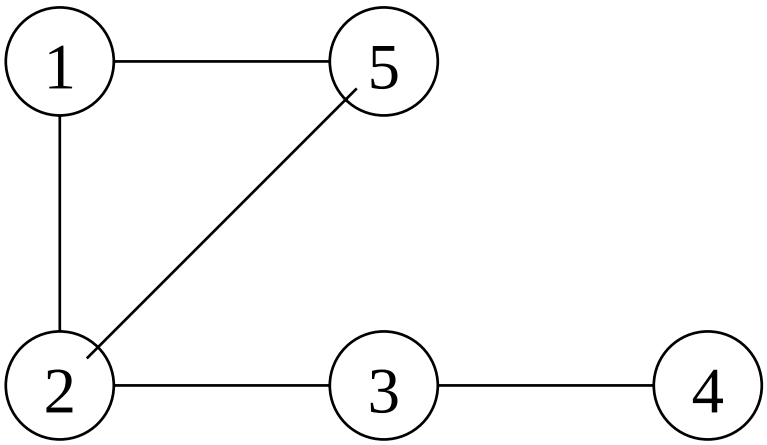


Figure 1: A graph with degree sequence 3, 2, 2, 2, 1.

Another example of a graphical sequence is $S_2 = 1, 4, 2, 2, 1$. (Try to generate a graph with degrees given by S_2.)

There are many non-graphical sequences; i.e., sequences that do not correspond to the degree sequence of any graph. For example $S_3 = 1, 2, 3, 1, 5$ is not graphical. We can determine this fact easily by noting that the length of the length of the sequence, and hence the number of nodes in the corresponding graph, should one exist, is 5. Therefore, no node can have a degree greater than 4 (because at most it can have edges connecting it to all the other nodes). Since S_3 contains a larger integer (5), it follows that S_3 is not graphical.

Your task is to write a program that determines whether a given sequence is graphical.

Hint: This problem is made easy by the following fact: A non-increasing sequence of integers $S = s_1, s_2, \ldots, s_k$, $k > 1$, is graphical if and only if $s_1 < k$ and $S' = s_2 - 1, s_3 - 1, \ldots, s_{s_1+1} - 1, s_{s_1+2}, s_{s_1+3}, \ldots, s_k$ is graphical.

Input Format

Your program must read its input from *standard input*. The input consists of a sequence of nonnegative integers separated by whitespace. Whitespace consists of a sequence of one or more characters chosen

from the following set: space, tab, and newline. Integers are represented in conventional decimal format (e.g., 42, 376). The input will contain at least one integer. The first integer N, in the input is the length of the sequence that is to be tested ($N \geq 0$). It is followed by N integers that form the sequence to be tested.

Output Format

Your program must write its output to *standard output*. If the input is a graphical sequence, your program should output the word `true` followed by a single newline character. If the input is not a graphical sequence, your program should output the word `false` followed by a single newline character. Your program must not produce any other output.

Examples

In the following examples, ␣ denotes the space character and each line is terminated by a single newline character (not visible).

Input 1:

```
6
1␣1␣0␣3␣0␣1
```

Output 1:

```
true
```

Input 2:

```
5
1␣␣␣␣␣␣4
␣␣␣2␣2␣␣␣1
```

Output 2:

```
true
```

Input 3:

```
6
1␣1␣0␣3␣1␣1
```

Output 3:

```
false
```

8 The Dwarfs' Casino

Gandalf the wizard needs to raise money to pay for all of those horses and weapons. So he stops by a local casino run by Gimli the Dwarf. talks with Gimli.

Gimli: We can't let you gamble! You're a wizard! You'll always win!
Gandalf: So what do you propose?
Gimli: After you declare how much you want to bet, we'll just say "You WIN" or "You LOSE."
Gandalf: But then you'll just tell me "You LOSE" all the time.
Gimli: We'll only say "You LOSE" a certain number of times.

With this in mind, they agree to the following game. There are three parameters (D, n, L). Initially Gandalf has D dollars (D is a positive integer). The following is repeated n times: Gandalf declares how much he will bet, a natural number b (which can range from 0 up to the total amount Gandalf currently has). After seeing the amount of the bet, the Dwarfs either say "You WIN" or "You LOSE".

If the Dwarfs says "You WIN" then the Dwarfs must pay Gandalf b dollars. On the other hand, if the Dwarfs says "You LOSE" then the Dwarfs take b dollars from Gandalf. However, the Dwarfs can only say "You LOSE" L times.

Here is an example. Suppose that Gandalf starts with $D = 12$ dollars, there are $n = 3$ rounds and $L = 1$ losses.

Round 1: Gandalf bets 6 dollar. The Dwarfs say "You WIN." Gandalf now has 18 dollars left.
Round 2: Gandalf again bets 6 dollars. The Dwarfs say "You LOSE." Gandalf now is back to 12 dollars.
Round 3: Gandalf now knows the Dwarfs cannot make him lose, so he bets all 12 dollars, and the Dwarfs say "You WIN." Gandalf wins 12 dollars, and walks away with his winnings of 24 dollars.

Could Gandalf had done better? It turns out that this is his best strategy, assuming that the Dwarfs play their best. (To verify this, try a different starting bet, and see what happens.) The Dwarfs are very clever and will play to minimize how much Gandalf wins. Lets help Gandalf be clever too!

Write a program which, given (D, n, L) outputs the following:

- What is Gandalf's best choice for his first bet?
- What is the most money Gandalf can win? (Assuming the Dwarfs play their best.)

Input Format

Your program will read in a series of lines, where each line contains three integers D, n, and L (where $D > 0$, $n > 0$, and $L \geq 0$), separated by spaces. (You may assume that the input is valid.)

Output Format

Your program will output Gandalf's initial bet and the maximum amount he can win from this bet. (See the examples below.)

Example

Input:	Output:
`12 3 1`	`Best initial bet: 6` `Maximum final profit: 24`
`18 2 1`	`Best initial bet: 6` `Maximum final profit: 24`
`12 1 0`	`Best initial bet: 12` `Maximum final profit: 24`
`12 2 1`	`Best initial bet: 4` `Maximum final profit: 16`

0 Triangle Flag (morning practice problem)

After constructing his tower, Saruman is considering changing his flag. After hearing your suggestions, Saruman decides that a simple triangle pattern will do fine. Your job is to help Saruman design his flag.

You begin with some object-oriented code for drawing pictures. A "picture" is a two-dimensional grid of colors, each of which contains a red, green and blue component in the range [0 - 255]. In these problems, a picture is defined by any class that implements the supplied Picture interface. The Picture interface's getWidth() and getHeight() methods defines the dimensions the picture, respectively. The getColor(int col, int row) method must specify the color of the picture at the specified position where (0, 0) specifies the top-left corner of the picture and the column number increases to the right and the row number increases downward.

Each color is defined by an instance of the supplied PictureColor class. A static utility method, PictureUtil.show() will display an instance of Picture on the screen. A supplied Image utility class can be used for creating a Picture given a jpeg image file.

Complete the code for class Triangle so it can generate a picture that forms a triangle pattern using the two specified colors with overall specified "width" and "height" dimensions. The first color is the background, the second color is the triangle. For simplicity you may assume that only triangles of equal width and height are created. A Triangle object created with the following parameters looks like:

new Triangle(256, 256, PictureColor.RED, PictureColor.YELLOW);

Input Format

The input will be two integers, specifying the height and width of the triangle flag. You may assume height and width are always the same.

Output Format

The program should produce a (upwards pointing) triangle pattern corresponding to the input parameters.

Example

The input "256 256" produces the triangle picture shown above.