**Task 5. Fork**

Luca opened a Python shell and typed `os.fork()`, which spawned a second shell. After that, each time Luca pressed any key, it would randomly go to one of the two shells. Each shell has an input string (shown in the terminal), which is edited by the key presses going to that shell. Additionally, Luca sees the terminal and thus he knows which shell's input string was affected when he presses a key.

His keyboard has $N$ keys with distinct characters on them and Backspace. When a character key press goes to a shell, the character is simply appended to the end of its input string. When a Backspace key press goes to a shell, the last character of its input string is erased. If the shell's input string is empty, nothing happens to it (though Luca still sees the Backspace went there). Each key press has probability $P$ of going to the left shell and probability $1 - P$ of going to the right one.

Luca wants to type some fixed string $a_1 a_2 \dots a_N$ consisting of $N$ distinct characters in both shells. He has already managed to type $L$ correct characters to the left shell and $R$ to the right one (i.e. the strings in the two shells are $a_1 a_2 \dots a_L$ и $a_1 a_2 \dots a_R$). For example, consider $P = 0.3$, $N = 2$ (the string could be ab), $L = 0$ and $R = 1$. A possible sequence of events is:

| Step | Key | Side | Left shell | Right shell |
|---|---|---|---|---|
| 0 | - | - | - | a |
| 1 | b | Right | - | ab |
| 2 | a | Right | - | aba |
| 3 | a | Left | a | aba |
| 4 | b | Right | a | abab |
| 5 | Backspace | Right | a | aba |
| 6 | Backspace | Left | - | aba |
| 7 | Backspace | Left | - | aba |
| 8 | Backspace | Right | - | ab |
| 9 | a | Left | a | ab |
| 10 | b | Right | a | abb |
| 11 | b | Left | ab | abb |
| 12 | Backspace | Right | ab | ab |

In total, typing ab to both shells took 12 key presses.

Let us define an incorrect character like so: a character in one of the two shells, which needs to be deleted at some point before typing the full string (and only that) to both shells. Luca has decided that he will never press Backspace, if there is no incorrect character in at least one of the shells. He has also decided that he will never press a key which will always result in an incorrect character. Luca is wondering what his optimal strategy would be under these constraints. In particular, he wants to know what is the minimum expected (average) number of key presses. Help Luca by writing a program `fork.cpp` which solves this problem.

**Input**

From the first and only line of the standard input, your program should read $P$, $N$, $L$ and $R$.

**Output**

On the first and only line of the standard output, your program should print the computed answer to (preferably) 12 digits of precision or more. You can use:

```
std::cout << std::setprecision(12) << ans << std::endl;
```

**Constraints**

$0 \leq L, R \leq N \leq 2 \times 10^7$
$0.1 \leq P \leq 0.9$

**Subtasks and scoring**

| Subtask | Points | $N \leq$ |
|---|---|---|
| 1 | 15 | 5 |
| 2 | 10 | 15 |
| 3 | 10 | 35 |
| 4 | 15 | 100 |
| 5 | 15 | 450 |
| 6 | 15 | 1500 |
| 7 | 15 | $10^6$ |
| 8 | 5 | $2 \times 10^7$ |

To get the points for a given subtask, your solution must successfully pass all tests in it and in all previous subtasks. To pass a test, your solution must print an answer with a relative error at most $10^{-8}$, i.e.:

$$\frac{|yourAns - trueAns|}{trueAns} \leq 10^{-8} \text{ (where } \frac{0}{0} = 0)$$

**Sample test**

| Input | Output |
|---|---|
| 0.3 2 0 1 | 16.7142857142857 |