RGPC '18 P2 - Performance Points

Time limit: 2.0s **Memory limit:** 64M

It's been a year of playing osu!, and **Kashif** is frustrated with the pp (performance point) system, since he can't get any new top scores. He wants to create his own score system, and wants your help to do so. To recall, osu! is an anime-themed rhythm game where circles appear sequentially on-screen, and players are required to click on them at just the right time to get points. If they hit the notes early or late, they get a reduced number of points, and if their timing is really off, they will miss and get no points. For the purpose of this question, refer to the following list of timings mapped to scores:

- Between 0ms and 50ms (absolute value, inclusive) of perfect timing: Excellent
- Between 51ms and 100ms: Great
- Between 101ms and 150ms: Good
- Between 151ms and 200ms: Bad
- Greater than 200ms: Miss

For example, hitting a note at 32ms (late) would be Excellent, and hitting a note at -128ms (early) would be Good. **Kashif** defines the following formula for calculating pp: $1.5a + b + c + 0.5d - e + 0.5f$, where a refers to the number of <Excellent>s, b the number of <Great>s, c the number of <Good>s, d the number of <Bad>s, e the number of <Miss>es, and f the length of the greatest combo (number of sequential notes clicked without a miss).

Given the number of notes in a song, **Kashif**'s previous score on the song, as well as the timings of his individual clicks during that playthrough, determine how much his score would increase by according to this new scoring system.

Input Specification

The first line of input will contain two numbers separated by a space: the integer N $(1 \le N \le 100\,000)$, representing the number of notes in the song, and the decimal number S $(0 \le S \le 100\,000)$, **Kashif**'s previous top score on the song. Each of the next N lines will each contain an integral timing value t_i $(-500 \le t_i \le 500)$, representing how early or late he hit the ith note, in milliseconds.

Output Specification

Output a single decimal number rounded to one decimal place, representing how much **Kashif**'s score would **increase** by.

Sample Input

```
5 3.0
35
-52
-140
-201
21
```

Sample Output

```
2.5
```