Lab Assignment 2 CS202 2024Spring

Problem 1. Vector Multiplication (20 pts)

Vector multiplication is an important operation in mathematics, and we would like you to implement the functionality to calculate the product of two vectors using RISC-V code.

Input Format

The input consists of several lines.

The first line contains a positive integer N $(1 \leq N \leq 100)$, which represents the dimension of the vectors.

The following N lines (from the 2nd line to the $N+1$ line) represent the content of the first vector, with each line containing a floating-point number.

The following N lines (from the $N+2$ line to the $2N+1$ line) represent the content of the second vector, with each line containing a floating-point number.

In this problem, you need to perform the calculations using the **float data type** in RV32. Therefore, when reading the input floating-point numbers, use the NO. 6 system call.

Output Format

Output a single floating-point number that represents the result of multiplying the two vectors.

Samples

Sample 1

Input

```
2
1.0
2.0
1.0
2.0
```

Output

```
5.0
```

Sample 2

Input

```
1
2.2
2.2
```

Output

```
4.84
```

Problem 2. Tiered Water Pricing (30 pts)

Tiered water pricing is a common form of water billing in cities.

Suppose a city has the following tiered water pricing rules:

- Rule 1: For water usage up to 15 tons, the price is 1 yuan per ton.
- Rule 2: For water usage between 15 and 20 tons, the first 15 tons follow Rule 1, and the remaining water usage is priced at 2 yuan per ton.
- Rule 3: For water usage exceeding 20 tons, the first 15 tons follow Rule 1, the portion between 15 and 20 tons follows Rule 2, and the remaining water usage is priced at 3 yuan per ton.

You need to write a program to assist users in calculating their water bill.

Input Format

The input consists of a single floating-point number X $(0 < X < 200)$, representing the water usage in tons.

In this problem, you need to perform the calculations using the **double data type** in RV32. Therefore, when reading the input floating-point number, use the NO. 7 system call.

Output Format

Output a single floating-point number O, representing the amount the user needs to pay for the water bill.

Samples

Sample 1

Input

```
5.0
```

Output

```
5.0
```

Sample 2

Input

```
25.0
```

Output

```
40.0
```

Problem 3. Root Finding with Bisection Method (50 pts)

The bisection method is a common approach for finding the roots of equations. In this problem, we would like you to use the bisection method to find one root of a cubic equation $f(x) = ax^3 + bx^2 + cx + d\ (a \neq 0)$.

The overall calculation process is as follows:

- We will provide you with two initial points, x_1 and x_2, for the bisection algorithm. These two points define the range of values where the root of the equation lies.
- Next, you need to iterate within this range and use the bisection method to approximate the root.
- In each iteration, you need to calculate $x_3 = \frac{x_1+x_2}{2}$. If $|f(x_3)| < 1e-6$, we consider that you have found the root of the equation and can exit the iteration process.
- If $|f(x_3)| \geq 1e-6$, you need to update the range.
- The range update rule is as follows: if $f(x_3) \times f(x_1) < 0$, set $x_2 = x_3$; otherwise, set $x_1 = x_3$.

Input Format

The input consists of 6 lines, each containing a floating-point number.

The first 4 lines represent the coefficients a, b, c, and d of the cubic equation, respectively.

The 5th and 6th lines represent the two initial points, x_1 and x_2, for the bisection algorithm. We guarantee that $x_1 < x_2$, $f(x_1) \neq 0$, $f(x_2) \neq 0$, and $f(x_1) \times f(x_2) < 0$.

In this problem, you need to perform the calculations using the **double data** type in RV32. Therefore, when reading the input floating-point numbers, use the NO. 7 system call.

Output Format

Output a single floating-point number x_0 that represents the root of the equation, satisfying the condition $f(x_0) < 1e-6$.

Samples

Sample 1

Input

```
2
3
-5
10
-10
0
```

output

```
-2.9333537817001343
```

Sample 2

Input

```
2
3
-5
8.5
-10
0
```

output

```
-2.880271226167679
```