KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Programavimo kalbų teorija (P175B124)

Laboratorinių darbų ataskaita

Atliko:
IFF-1/6 gr. studentas
Lukas Kuzmickas
2023 m. vasario 28 d.
Priėmė:
Doc. Sajavičius Svajūnas

KAUNAS 2023

TURINYS

1. C++ arba Ruby (L1)

1.1. Darbo užduotis

You are to determinate X by given Y, from expression $X = \sqrt{Y}$

Input

The first line is the number of test cases, followed by a blank line.

Each test case of the input contains a positive integer Y ($1 \leq Y \leq 10^{1000}$), with no blanks or leading zeroes in it.

It is guaranteed, that for given Y, X will be always an integer.

Each test case will be separated by a single line.

Output

For each test case, your program should print X in the same format as Y was given in input.

Print a blank line between the outputs for two consecutive test cases.

Sample Input

```
1

7206604678144
```

Sample Output

```
2684512
```

1 pav. Užduoties pavyzdys iš rinkinių.

Užduotis labai paprasta, turime kintamuosius Y ir X. Turime įvedimo ir išvedimo failus, Y saugomas įvedimo faile, X išvedamas į išvedimo failą. Mes paskaičiuojame X reikšmę, pagal formulę $X = \sqrt{Y}$. Pradžioje įvedimo failo nurodome atvejų kiekį ir pačias Y reikšmes. X išvedamas su eilučių tarpais, jeigu turime daugiau nei vieną duomenų atvejį.

1.2. Programos tekstas

Program.cpp

```
//AUTHOR Lukas Kuzmickas IFF-1/6
#include <iostream>
#include <fstream>
#include <string>
#include <chrono>
#include <vector>
#include <iostream>
#include <sstream>
using namespace std;

/// <summary>
/// Input class for file input operations
/// </summary>
static class Input {
public:
    Input(const string& filename) : infile(filename) {}
    /// <summary>
    /// Method to read from string filename to an integer data type
    /// </summary>
    /// <returns>integer</returns>
    int readInt() {
        int n;
        infile >> n;
        return n;
    }

    /// <summary>
    /// Method to read a file to a string
    /// </summary>
    /// <returns>string</returns>
    string readString() {
        string s;
        infile >> s;
        return s;
    }

    /// <summary>
    /// Method to check if file isn't empty (End of file)
    /// </summary>
    /// <returns>true or false</returns>
    bool eof() {
        return infile.eof();
    }

private:
    ifstream infile;
};


/// <summary>
/// Output class for file output operations
/// </summary>
static class Output {
public:
    Output(const string& filename) : outfile(filename) {}

    /// <summary>
    /// Write function for writing output to a data file
    /// </summary>
```

```
    /// <param name="s">string to write</param>
    void write(const string& s) {
        outfile << s;
    }

    /// <summary>
    /// Write function for writing output with new line to a data file
    /// </summary>
    /// <param name="s">string value</param>
    void writeLine(const string& s) {
        outfile << s << endl;
    }
private:
    ofstream outfile;
};


/// <summary>
/// Data class for Y object
/// </summary>
class Y {
private:
    string y_str;
public:
    Y(const string& str) : y_str(str) {}
    /// <summary>
    /// Putting all Y value to a vector digit list
    /// </summary>
    /// <returns>vector value digit list</returns>
    vector<int> digits() const {
        vector<int> y_digits(y_str.size());
        for (int i = 0; i < y_str.size(); i++) {
            //ASCI code convertion/deletion
            y_digits[i] = y_str[i] - '0';
        }
        return y_digits;
    }
};

/// <summary>
/// Data class for X object
/// </summary>
class X {
public:
    X(int value) : value(value) {}
    /// <summary>
    /// Convert X value to string
    /// </summary>
    /// <returns>string</returns>
    string to_string() const {
        return std::to_string(value);
    }
private:
    int value;
};


/// <summary>
/// Class for data operations
/// </summary>
class TaskUtils {
public:
    /// <summary>
    /// Method for converting from string to double
    /// </summary>
    /// <param name="str">given string</param>
```

```
    /// <returns>double value</returns>
    static double stringToDouble(const std::string& str) {
        double result;
        std::istringstream stream(str);
        stream >> result;
        return result;
    }
    /// <summary>
    /// Method for getting root square of given value
    /// </summary>
    /// <param name="Y">Given value</param>
    /// <returns>root square</returns>
    static double squared(double Y)
    {
        return sqrt(Y);
    }
};
int main() {
    //INPUT for Y
    Input in("input_y.txt");
    //OUTPUT for X
    Output out("output_x.txt");
    using namespace std::chrono;
    //chrono objects for time calculation
    time_point<system_clock> start, end;
    start = system_clock::now();

    //number of test cases
    int t = in.readInt();
    while (!in.eof() && t > 0) {
        Y y(in.readString());
        vector<int> y_digits = y.digits();
        string y_str;
        for (int i = 0; i < y_digits.size(); i++) {
            y_str += to_string(y_digits[i]);
        }
        double Y_double = TaskUtils::stringToDouble(y_str);
        X x = TaskUtils::squared(Y_double);
        out.writeLine(x.to_string());
        if (!in.eof()) {
            out.writeLine("");
        }
        t--;
    }
    end = system_clock::now();
    duration<double> elapsed_seconds = end - start;
    cout << "Total time elapsed " << elapsed_seconds.count() << "s\n";
    return 0;
}
```

1.3. Pradiniai duomenys ir rezultatai

Pirmieji duomenys:

input_y.txt
```
2
169
81
```

output_x.txt
```
13

9
```

Antrieji duomenys:

input_y.txt
```
10
9000000
31382404
40401
324
9
25
16
4
81
64
```
output_x.txt
```
3000

5602

201

18

3

5

4

2

9

8
```

2. L2

2.1. Darbo užduotis

2.2. Programos tekstas

2.3. Pradiniai duomenys ir rezultatai

3. L3

3.1. Darbo užduotis

3.2. Programos tekstas

3.3. Pradiniai duomenys ir rezultatai

4. L4

4.1. Darbo užduotis

4.2. Programos tekstas

4.3. Pradiniai duomenys ir rezultatai