aLLIotaLP

Чемпионат по программированию: Бэкенд-разработка - Квалификация

	3 ноя 2019, 17:51:29
старт:	2 ноя 2019, 20:02:59
финиш:	3 ноя 2019, 01:02:59
длительность:	05:00:00
начало:	14 окт 2019, 12:00:00
конец:	20 окт 2019, 23:59:59

B. Калькулятор стоимости поездки

	Все языки	Python 3.7.3	Python 2.7
Ограничение времени	3 секунды	65 секунд	65 секунд
Ограничение памяти	64Mb	256Mb	256Mb
Ввод	стандартный ввод или input.txt		
Вывод	стандартный вывод или output.txt		

Нужно рассчитать стоимости поездок по заданной формуле. Каждая поездка характеризуется K целочисленными параметрами.
Формула задается в обратной польской записи.
Допустимые операции:

- **+ -** – сложение и вычитание;
- *** /** – умножение и целочисленное деление;
- **< =** – сравнение;
- **?** – условный оператор. Если первый аргумент истина — возвращает второй аргумент, иначе — третий;

В формуле также используются переменные [a-z] и целые числа от -10^9 до 10^9.

Можно считать, что результаты всех операций в формуле не превышают 10^9 по абсолютному значению. Результат операций сравнения используется только в качестве аргумента условного оператора.

Формат ввода

На первой строке одно число $1 \le K \le 26$ — количество переменных.

На второй строке записана формула расчета цены (не более $3 \cdot 10^4$ элементов). Все элементы разделены пробелами.

На третьей строке $1 \le N \le 10^4$ — число тестов.

Следующие N строк содержат по K целых чисел ($-10^9 \le v \le 10^9$) — значения переменных в алфавитном порядке.

Формат вывода

N строк, содержащих по одному целому числу — результаты подстановки каждого набора значений. Гарантируется, что результат выражения конечен и определен.

Пример 1

Ввод	Вывод

Ввод	Вывод
1 a 2 2 + * 2 2 3	8 12

Пример 2

Ввод	Вывод
2 a b < 5 14 ? 2 10 5 5 10	14 5

Язык GNU c++17 7.3

Набрать здесь | Отправить файл

```cpp
#include <memory>
#include <sstream>
#include <string>

#include <stack>
#include <vector>

#include <iostream>

using operand_t = int64_t;

struct IOperator {
    virtual operand_t Perform(const operand_t* vars) const = 0;
    virtual ~IOperator() = default;
};

using IOperatorPtr = const IOperator*;

class Calculator {
    IOperatorPtr _expressionTree;
    std::vector<std::unique_ptr<IOperator>> _allOperations;
public:
    explicit Calculator(const std::string& formula);
    operand_t Calculate(const operand_t* vars);
};

int main() {
    std::string kStr, formula, nStr;

    getline(std::cin, kStr);
    getline(std::cin, formula);
    getline(std::cin, nStr);

    auto k = std::stoul(kStr);
    auto calculator = Calculator(formula);
    auto n = std::stoul(nStr);
```

Отправить

Предыдущая | Следующая