Zadanie $\beta*$
Kalkulator

Dziś w drzwiach firmy Bajtomania kolejny raz stanął znany Ci już Teodozjusz Malkontent. Uradowany poprzednim programem złożył kolejne zamówienie. Bada on niezwykle ciekawe (w jego mniemaniu...) fizyczne zjawiska i wykonuje całą masę obliczeń na ogromnych liczbach. Aby ułatwić sobie prace, potrzebuje kalkulatora, działającego na bardzo dużych liczbach i realizującego podstawowe operacje matematyczne. W szczególności potrzebuje on następujących operatorów:

1. `+ x y`, realizujący operację `x += y`
2. `- x y`, realizujący operację `x -= y`
3. `* x y`, realizujący operację `x *= y`
4. `< x y`, zdefiniowany jako

$$\begin{cases} -1 & \text{gdy } x < y \\ 0 & \text{gdy } x = y \\ 1 & \text{gdy } x > y \end{cases}$$

Wejście

Pierwsza linia wejścia zawiera liczbę całkowitą z ($1 \leqslant z \leqslant 2 \cdot 10^9$) – liczbę operacji do wykonania. Opis operacji składa się z jednej linii.

Pierwszy znak tej linii oznacza rodzaj wykonywanej operacji, następnie znajdują się dwa napisy oddzielone spacją, reprezentujące liczby. Można założyć, że $x, y \leqslant 10^{100}$.

Wyjście

Dla każdej operacji wypisz jedną linię zawierającą wynik.

Dostępna pamięć: 2MB
Wymagany język: C++

Przykład

Dla danych wejściowych:

```
11
+ 1210 9880
- 1210 9880
- 1120 335
* 1210 110
< 1210 9880
< 1 1
< 18210 9880
- -100 9200
-  100 -9200
+  199 -9200
+  -199 9200
```

Poprawną odpowiedzią jest:

```
11090
-8670
785
133100
-1
0
1
-9300
9300
-9001
9001
```

Zadanie należy zrealizować poprzez zaimplementowanie opisanej poniżej klasy `number`.

- Klasa posiada trzy pola prywatne: tablicę pamiętającą cyfry liczby, pole pamiętające liczbę cyfr w liczbie oraz pole pamiętające znak liczby.
- Klasa posiada posiada konstruktor domyślny ustawiający liczbę na wartość 0 oraz konstruktor z jednym argumentem typu `string`. Przykłady wywołania:

  ```
  number a("12300");
  number b("-12300");
  number c;
  ```

- Klasa udostępnia następujące metody operatorowe:
  1. `number operator+=(number& a)` – dodanie liczby a.
  2. `number operator-=(number& a)` – odjęcie liczby a.
  3. `number operator*=(number& a)` – pomnożenie liczby przez a.
- Klasa jest zaprzyjaźniona z trzema operatorami:
  1. `int operator<(number& A, number& B)` – zwraca wynik porównania liczb A i B. Operator zwraca -1 jeśli dana liczba A jest mniejsza od B, 0 gdy są równe oraz 1 gdy liczba A jest większa od B.
  2. `ostream& operator<<(ostream &stream, number &A)` – wypisuje liczbę A do strumienia `stream`.
  3. `istream& operator>>(istream &stream, number &A)` – wczytuje liczbę A ze strumienia `stream`.
- Klasa może zawierać dodatkowe pola lub metody.

Definicję klasy wraz z definicjami wszystkich metod i operatorów należy umieścić w pliku z rozszerzeniem `.h`. Tak przygotowany plik należy wysłać na Satori. Zostanie on skompilowany wraz z poniższym plikiem zawierającym funkcję `main`.

```
#include<iostream>
#include<string>
using namespace std;
#include "solution.h"

int main()
{
   ios_base::sync_with_stdio(false);
   string s;

   number C;
   number D("123");
   cout << C << " " << D << endl;

   int z;
   cin >> z;
   while (z--)
   {
      number A;
      number B;
      cin >> s >> A >> B;

      switch (s[0])
      {
         case '+' : A += B;  cout << A << endl; break;
         case '-' : A -= B;  cout << A << endl; break;
         case '*' : A *= B;  cout << A << endl; break;
         case '<' : cout << (A < B) << endl;
      }
   }
   return 0;
}
```