

Космические гонки. 11 классы

Винтик и Шпунтик решили организовать гонки на ракетах. В гонках решили принять участие почти все коротышки, включая Незнайку и Селёдку. В квалификационном старте побеждает тот коротышка, чья ракета наберёт самую большую скорость. Наблюдательный пункт Винтика расположен в точке Лагранжа L1 системы Земля-Луна. Для замера скорости Винтик делает два сферических снимка пространства вокруг точки наблюдения, а затем вычисляет перемещение ракет участников. Тот коротышка, чья ракета пролетела наибольшее расстояние, побеждает в квалификационном старте. Если ракета присутствует на одном снимке из двух, она дисквалифицируется из гонки.

Формат ввода: На стандартном потоке ввода задаются два сферических снимка пространства вокруг точки наблюдения. Снимки разделяются строкой ——. Каждая ракета описывается своим именем, склонением, прямым восхождением и расстоянием в км. Каждое описание находится на отдельной строке.

Формат вывода: На стандартный поток вывода напечатайте максимальное перемещение ракеты в километрах. Число выводите в формате с 10 значащими цифрами. Если во входных данных нет ракет, либо все ракеты дисквалифицированы, то выведите число -1.

Ввод примера №1:

```
Незнайка -16°42'58.02" 06h45m8.92s 1.66532e6
Селёдка -09°27'29.7312" 03h32m55.84496s 1043
Фуксия +89°15'50.8" 02h31m49.09s 1000001.45
--
Незнайка -16°42'03" 06h40m16s 1.66e6
Селёдка -05°18'44" 03h32m55.84496s 1200
```

Вывод примера №1:

34339.0462

Код возможного решения

```
#include <iostream>
#include <string>
#include <cstdio>
#include <cctype>
#include <cstring>
#include <cmath>
#include <map>

using namespace std;

constexpr double PI = 3.1415926535897932384626433832795028841971;
constexpr double DECL_SEC_TO_RAD = .000004848136811095359935899141;
constexpr double DECL_MIN_TO_RAD = .000290888208665721596153948461;
constexpr double DECL_GRAD_TO_RAD = .017453292519943295769236907684;
constexpr double RA_SEC_TO_RAD = .000072722052166430399038487115;
constexpr double RA_MIN_TO_RAD = .004363323129985823942309226921;
constexpr double RA_HOUR_TO_RAD = .261799387799149436538553615273;

struct Star
{
    string name;
    double x{}, y{}, z{};

    Star() = default;
    Star(string nn, double xx, double yy, double zz) :
```

```

        name(std::move(nn)),
        x{xx},
        y{yy},
        z{zz}
    {
    }
    Star(const std::string &buf);
};

Star::Star(const std::string &buf)
{
    char *name = nullptr;
    double dg{}, dm{}, ds{}, rah{}, ram{}, ras{}, dist{};
    sscanf(buf.c_str(), "%ms %lf°%lf' %lf" %lfh%lfm%lfs %lf",
           &name, &dg, &dm, &ds, &rah, &ram, &ras, &dist);
    Star::name = std::string(name);
    double ra = rah * RA_HOUR_TO_RAD + ram * RA_MIN_TO_RAD + ras * RA_SEC_TO_RAD;
    double decl = 0;
    if (signbit(dg)) {
        decl = dg * DECL_GRAD_TO_RAD - dm * DECL_MIN_TO_RAD - ds * DECL_SEC_TO_RAD;
    } else {
        decl = dg * DECL_GRAD_TO_RAD + dm * DECL_MIN_TO_RAD + ds * DECL_SEC_TO_RAD;
    }
    z = sin(decl) * dist;
    double prj = cos(decl);
    x = prj * cos(ra) * dist;
    y = prj * sin(ra) * dist;
}

int main()
{
    string buf;
    map<string, Star> stars1;
    map<string, Star> stars2;

    while (getline(cin, buf)) {
        size_t l = buf.length();
        while (l > 0 && isspace((unsigned char) buf[l-1])) --l;
        buf.resize(l);
        if (buf == "--") {
            break;
        }
        Star ss(buf);
        stars1.insert({ ss.name, ss });
    }

    while (getline(cin, buf)) {
        size_t l = buf.length();
        while (l > 0 && isspace((unsigned char) buf[l-1])) --l;
        buf.resize(l);
        Star ss(buf);
        stars2.insert({ ss.name, ss });
    }

    double dist = -1;

```

```

for (const auto &[n, s1] : stars1) {
    if (auto it = stars2.find(n); it != stars2.end()) {
        auto dx = s1.x - it->second.x;
        auto dy = s1.y - it->second.y;
        auto dz = s1.z - it->second.z;
        auto dd = hypot(dx, dy, dz);
        if (dist < 0 || dd > dist) {
            dist = dd;
        }
    }
}
printf("%.10g\n", dist);
}

```