Problem A. Большие данные

Input file:	*none*
Output file:	*текстовый ввод*
Time limit:	1 секунда
Memory limit:	512 мебибайт

Так как при переходе к каждой следующей единице «двоичные» величины увеличиваются в 2^10 раз, а «десятичные» в 10^3, то фактически требуется найти разность между 2^{10^4} и 10^{3^4}, или же между 2^{40} и 10^{12}.

Её можно вычислить как с помощью компьютера (калькулятора, приложения «Калькулятор»), так и вручную, возведя приведённое значение 2^{20} в квадрат и вычтя 10^{12}. Так как рассматриваются диски объёмом 4 терабайта и 4 тебибайта, то для получения итогового ответа разность требуется умножить на 4.

Problem B. Тридевятое царство

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

За $N-2$ дороги справиться легко — например, соединив все города, не имеющие выхода к морю (а таких будет $N-2$) с каким-то одним из двух городов, имеющих выход к морю.

Покажем, что если каждый город имеет выход к морю, то дорог не менее $N-2$. Для этого будем двигаться от моря к каждому городу. Все города, до которых можно дойти, проехав одну дорогу, обозначим за 1, все города, которые ещё не имеют обозначений и до которых можно дойти, проехав две дороги, за 2 и так далее. Очевидно, что если все города соединены с морем, то рано или поздно каждый город получит обозначение. При этом дорога, по которой в этот город пришли, до этого не проходилась (иначе бы этот город уже имел обозначение). Таким образом, каждому городу не у моря однозначно соответствует дорога, и тем самым дорог не менее $N-2$.

Таким образом, программа должна считывать N и выводить число $N-2$.

```
#include <stdio.h>

main()
{
    int n;
    scanf("%d", &n);
    printf("%d\n", n - 2);
    return 0;
}
```

Problem C. Частотный словарь

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 секунда
Memory limit:	64 мебибайта

Для решения задачи заметим, что количество вхождений какой-либо буквы в текст равно сумме количества вхождений этой буквы во все слова. Таким образом, требуется прочитать все строки словаря и для каждой буквы текущего слова прибавить к счётчику количества вхождений этой буквы (таких счётчиков, очевидно, всего 26) кратность текущего слова. После чего из 26 счётчиков стандартной (или самостоятельно написанной) процедурой выбрать наибольший.

```
#include <stdio.h>
#include <cstring>

int letters[26];

main()
{
    int N, q, max = 0;
    char buf[1000];
    for (int i = 0; i < 26; i++)
        letters[i] = 0;
    scanf("%d", &N);
    for (int j = 0; j < N; j++) {
        scanf("%s %d", buf, &q);
        for (int i = 0; i < strlen(buf); i++)
            letters[buf[i] - 'a'] += q;
    }

    for (int i = 0; i < 26; i++)
            if (letters[i] > max)
            max = letters[i];
    printf("%d\n", max);
    return 0;
}
```

Problem D. Треугольник

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Обозначим гипотенузу за c, а второй катет за b. Из теоремы Пифагора $c^2 = p^2+b^2$, то есть $p^2 = c^2-b^2$, или $p^2 = (c+b)(c-b)$. Разложить квадрат простого числа на целые множители можно двумя способами $p \cdot p$ (в этом случае $c+b = c-b$ и $b = 0$, что невозможно по условию задачи) или $1 \cdot p^2$. Получаем систему уравнений $+b = p^2$ и $c-b = 1$. Решая её, получаем, что $c = (p^2-1)/2$.

Таким образом, если $p = 2$, целочисленных прямоугольных треугольников со стороной p не существует. Иначе длина гипотенузы восстанавлисявется однозначно по выведенной выше формуле.

Написание программы, реализующей это решение, не должно вызвать затруднений (один условный оператор, а затем вычисления по формуле). Стоит обратить внимание, что ответ не помещается в 32-битный тип данных, так что для хранения ответа следует использовать 64-битное целое.

Пример решения задачи на языке C.

```
#include <stdio.h>

main()
{
    long long p;
    scanf("%lld\n", &p);
    if (p == 2LL)
printf("-1\n");
    else
printf("%lld\n", (p * p + 1) / 2LL);
    return 0;
}
```

Problem E. Лара и странный барельеф

Input file:	*none*
Output file:	*текстовый ввод*
Time limit:	1 second
Memory limit:	64 mebibytes

К ответу на задачу можно придти несколькими способами.

Можно заметить в слове “LINUX”, которе имеет значение 59, подстроку “LIX” — римскую запись числа 59.

Можно обратить внимание на то, что в словах со значением, превосходящих 1000, содержится буква ‘M’, а в слове “PROGRAMMING” таких букв две. Таким образом, сопоставить букву ‘M’ и 1000, после чего догадаться об использовании римских чисел в данной задаче.

Алгоритм шифровки выглядит так: все буквы, не являющиеся римскими цифрами, отбрасываются. Если получившееся римское число некорректно (например, IMIC в слове “INFORMATICS”, выводится сообщение об ошибке. Иначе выводится его значение или 0, если ни одной цифры не осталось.

При проведении соответствующих операций над словом “MEDALIST” получается MDLI, или 1551.