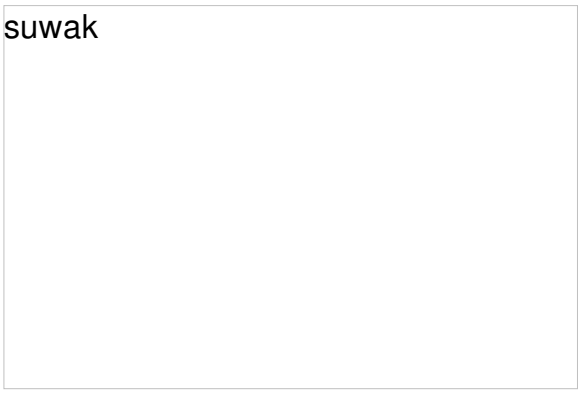
Suwak logarytmiczny, hypot i long double



Jeżeli wiesz do czego służy, to i tak prawdopodobnie nie masz biegłości w posługiwaniu się suwakiem logarytmicznym [tak jak i liczydłami czy abakusem]. Jedną z wielu zalet tych urządzeń jest zapamiętywanie pośredniego [końcowego] wyniku aż do momentu wykasowania go, zupełnie bez użycia baterii, czy akumulatora jak to się dzieje w większości dzisiejszych urządzeń. Inną zaletą suwaka jest brak błędów przepełnienia overflow i underflow. Wadą wszystkich tych urządzeń jest skończona dokładność liczenia, suwak to tylko dwie cyfry znaczące, a w komputerze, używając typu **long double**, mamy około dziesięć razy więcej tych cyfr.

Policz jak najszybciej i z jak największą dokładnością, końcową odległość [po n iteracjach], pomiędzy dwoma zmieniającymi swoje położenie punktami P_1 i P_2.

Ta końcowa odległość równa jest odległości policzonej algorytmem:

powtarzaj n razy:

wynik = odl(P1,P2)

gdy n > 0

[P1] = [P1] * wynik, [P2] = [P2] * wynik //mnożymy współrzędne x, y obu punktów

gdy n < 0

[P1] = [P1] / wynik, [P2] = [P2] / wynik //dzielimy współrzędne x, y obu punktów

Wynik wypisz w zaokrągleniu z dokładnością do d cyfr znaczących. Współrzędne punktów i końcowy wynik obliczeń są tak dobrane, by mieściły się w zakresie typu **long double**. Wartości n i d są z przedziału: **0 < |n| < 100, 0 < d < 20.**

Wejście

Nieokreślona ilość testów. Każdy test to jedna linia zawierające następujące dane: najpierw cztery współrzędne $\mathbf{x_1, y_1, x_2, y_2}$, a następnie ilość iteracji **n** i ilość cyfr znaczących do wydrukowania **d**.

Wyjście

Dla każdego testu należy w oddzielnej linii wypisać odpowiedź - obliczoną odległość.

Przykład

Wejście:
2 2 1 1 -1 3
1 1 2 2 1 1

0 0 -1e-10 1e-10 -1 4

Wyjście:
1.41
1
1.414e-10

Podpowiedź:

We wzorcówce użyto funkcji cout. Aby uzyskać taki sam format wydruku wyników użyj:

```
cout << setprecision(d) << wynik << endl;
```