上機考 Computer-Based

考試時間 Test time 15:51:00 pm to 17:17:00 pm. 以 NTU COOL 時鐘為準 Using the clock of NTU COOL as official time. 在 NTU COOL 作業區上傳程式碼 Upload cpp files to NTU COOL assignment section. 每筆測資一分 One point per testing case. 題號數量即為測資數量 A question index in a pair of square brackets represents one testing case. 可上網及查書 You may loop up information online or in books. 可用人工智慧 AI is allowed. 禁止合作討論 No collaboration or discussion. 禁止求助真人、論壇、社群媒體等 Do not seek help from forums, social media, chatting apps, etc. 舉手問問題 Raise your hand to ask questions.

[65] [66] [67] [68] [69] [70] [71] [72] [73] [74] On Jan 5, 2030, you borrowed L TWD, $1 \le L \le 10^{16}$, from a bank to invest in chip manufacturing. On the 10th day of each month, the bank adds a 0.2% interest to your outstanding balance, rounded up to the nearest 0.1 TWD. On the 15th day of each month, you pay the bank P TWD. You wish to repay the loan by Dec 20, 2040 while minimizing your monthly payment. But you are afraid of another financial crisis, which might cause a raise in the interest rate. Your financial advisor said that the worst that could happen is that the interest rate might rise to 0.3% for six months; but when will that happen is hard to predict. You trust this advice as it is from a graduate of the College of Management, National Taiwan University, renowned for its rigorous curriculum and strong reputation for producing top business leaders and financial experts.

Write a function that takes `long long L` as input and returns the least possible `long long P`. Binary search (or anything sub-linear) is necessary. You may start from the following.

```
#include <iostream>
using namespace std;
bool simulate_payment(long long L, long long P) {
}
long long calculate_min_payment(long long L) {
}
int main() {
    long long TestingCases[5] = {1000, 1000, 10000, 100000, 1000000};
    for (int ndx = 0; ndx < 5; ndx ++) {
        long long L = TestingCases[ndx];
        long long P = calculate_min_payment(L);
        cout << "Loan:␣" << L << "␣-->␣Payment:␣" << P << "\n";
    }
}
/*
 * 1000 --> 9
 * 10000 --> 87
 * 100000 --> 868
 * 1000000 --> 8678
 * 10000000 --> 86780
 * 100000000 --> 867791
 * 1000000000 --> 8677909
 * 10000000000 --> 86779082
 * 100000000000 --> 867790813
 * 1000000000000 --> 8677908125
 * 10000000000000 --> 86779081241
 * 100000000000000 --> 867790812405
 * 1000000000000000 --> 8677908124043
 * 10000000000000000 --> 86779081240425
 */
```

[75] [76] [77] [78] [79] [80] [81] [82] [83] [84] This is what a high-end CPU looks like from side view.

```
int Fins[15][10] = {
    {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
    {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
    {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
    {2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2},
    }
```

Here 2 is the actual CPU that generates heat, 1 is the fins, and `0` is the air. CPU generates heat at a rate of g°C per second per unit length squared. Within CPU, heat transfers at a rate of $(\delta/50)$°C per second per unit length, where δ is the temperature difference. At the contact surface between the CPU and the fins, heat transfers at a rate of $(\delta/20)$°C per second per unit length. Within the fin, heat transfers at a rate of $(\delta/10)$°C per second per unit length. At the contact surface between the fins and the air, heat transfers at a rate of $(\delta/100)$°C per second per unit length. And the air is always at 30°C.

Now you are asked to write a program that takes `double g` as input and outputs the temperature of the hottest point of the CPU after a long long time, so long that the temperature changes by at most $(1/1000000)$°C per second.

```
    #include <iostream>
    using namespace std;
    int Fins[10][15] = {
        {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
        {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
        {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
        {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
        {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
        {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
        {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
        {0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
        {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1},
        {2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2}
    };
    void SimulateOneSecond(int Temper[10][15], g){
    }
    double SimulateCpuTemperature(double g) {
    }
    int main() {
        double TestingCases[10] = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0};
        for (int ndx = 0; ndx < 10; ndx ++) {
            double g = TestingCases[ndx];
            double hottest = SimulateCpuTemperature(g);
            cout << "g:␣" << g << "␣-->␣Hottest:␣" << hottest << "\n";
        }
        return 0;
    }
    /*
     * 0.1 --> 42.52
     * 0.2 --> 55.0403
     * 0.3 --> 67.5607
     * 0.4 --> 80.0811
     * 0.5 --> 92.6014
     * 0.6 --> 105.122
     * 0.7 --> 117.642
     * 0.8 --> 130.163
     * 0.9 --> 142.683
     * 1.0 --> 155.203
     */
```

[85] [86] [87] [88] [89] [90] [91] [92] [93] [94] In quantum information theory, a qubit is represented by a density matrix

$$\rho = \begin{bmatrix} p & r+si \\ r-si & q \end{bmatrix} \in \mathbb{C}^{2\times 2},$$

where i is the imaginary unit. In C++, we represent it as

```
    struct Qubit {
        double p, q, r, s;
    };
```

There are four operations. The measurement operation (denoted by `M`) turns ρ into

$$\begin{bmatrix} \frac{p}{p+q} & 0 \\ 0 & \frac{q}{p+q} \end{bmatrix} \in \mathbb{C}^{2\times 2}.$$

The NOT gate (denoted by N) turns ρ into

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} p & r+si \\ r-si & q \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

The Hadamard gate (denoted by H) turns ρ into

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} p & r+si \\ r-si & q \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

The Fujisan gate (denoted by F) turns ρ into

$$\begin{bmatrix} 3 & 4 \\ -4 & 3 \end{bmatrix} \begin{bmatrix} p & r+si \\ r-si & q \end{bmatrix} \begin{bmatrix} 3 & -4 \\ 4 & 3 \end{bmatrix}$$

Your job is to write a program that

- initializes ρ as the matrix $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$,
- takes as input a string consisting of M, N, H, and F, and
- outputs $p - q$ after performing those operations.

The operations are interpreted from left to right. That is, if the input is NHM, you should

- apply the NOT gate,
- apply the Hadamard gate,
- perform a measurement.

You may start from here.

```cpp
#include <iostream>
using namespace std;
struct Qubit {
    double p, q, r, s;
};
Qubit initialize() {
}
void apply_measurement(Qubit &Q) {
    // cout << "M " << Q.p << " " << Q.q << " " << Q.r << " " << Q.s << endl;
}
void apply_not(Qubit &Q) {
    // cout << "N " << Q.p << " " << Q.q << " " << Q.r << " " << Q.s << endl;
}
void apply_hadamard(Qubit &Q) {
    // cout << "H " << Q.p << " " << Q.q << " " << Q.r << " " << Q.s << endl;
}
void apply_fujisan(Qubit &Q) {
    // cout << "F " << Q.p << " " << Q.q << " " << Q.r << " " << Q.s << endl;
}
void process_operations(Qubit &Q, string &operations) {
}
int main() {
    string TestingCases[12] = {
        "HM", "NM", "FM",
        "HHM", "HNM", "HFM",
        "NHM", "NNM", "NFM",
        "FHM", "FNM", "FFM"
    };
    for (int ndx = 0; ndx < 12; ndx ++) {
        string operations = TestingCases[ndx];
        Qubit Q = initialize();
        process_operations(Q, operations);
```

```
            cout << operations << "␣-->␣" << Q.p - Q.q << endl;
        }
    }
    /*
     * HM --> 0
     * NM --> -1
     * FM --> -0.28
     * HHM --> 1
     * HNM --> 0
     * HFM --> 0.96
     * NHM --> 0
     * NNM --> 1
     * NFM --> 0.28
     * FHM --> -0.96
     * FNM --> 0.28
     * FFM --> -0.8432
     */
```

[95] For these bonus problems, the only method I know is brute force, and I do not even know if a solution exists: Find an operation string such that $p - q$ is in the range $1/\pi \pm 1/10$.

[96] Make $p - q$ in the range $1/\pi \pm 1/100$.

[97] Make $p - q$ in the range $1/\pi \pm 1/1000$.

[98] Make $p - q$ in the range $1/\pi \pm 1/10000$.

[99] Make $p - q$ in the range $1/\pi \pm 1/100000$.

[100] Make $p - q$ in the range $1/\pi \pm 1/1000000$.

Upload your best string (in a txt file, not cpp) to NTU COOL.

[101] `frag1.txt`: Upload a txt file containing a string of length 900, and it should be a substring of the true answer.

[102] `frag2.txt`: Same rule as above.

[103] `frag3.txt`: Same rule as above.

[104] `frag4.txt`: Same rule as above.

[105] `frag5.txt`: Same rule as above.

[106] `frag6.txt`: Same rule as above.

[107] `frag7.txt`: Same rule as above.

[108] `relative8.txt`: Upload a txt file containing a string like this: For relative8.txt, the first line is the stranger.

[109] `relaive9.txt`

[110] `relaive10.txt`

[111] `sid11.txt`

[112] `sid12.txt`

[113] `sid13.txt`

[114] `sid14.txt`