答案卷 Answer Sheet

考試時間 Test time 15:51:00 pm to 18:18:00 pm. 以教室後方時鐘為準 Using the clock at the back of the classroom as official time. 在本頁作答 Answer on this page. 每題一分 One point per problem. 全對才給分 No partial credit. 不倒扣 No penalty for wrong answers. 禁止外部資源（比照學測、指考） No external resources. 舉手問問題 Raise your hand to ask questions.

[Name] [Student ID]

[0] [1] [2] [3] [4] [5] [6]

[7] [8] [9] [10] [11] [12] [13] [14]

[15] [16] [17] [18] [19]

[20] [21] [22] [23] [24] [25] [26]

[27] [28] [29] [30] [31] [32] [33] [34]

[35] [36] [37] [38] [39] [40] [41] [42] [43] [44]

[45]

[46]

[47] [48]

[49] [50]

[51] [52] [53]

[54]

[55] [56] [57] [58] [59] [60] [61] [62]

[63]

[64]

[0] Select the only one correct hello world.

(A)
```
#include <iostream>
int main() {
    cout << "helloworld";
}
```

(B)
```
#include <iostream>
int main() {
    std::cout << "helloworld";
}
```

(C)
```
#include <iostream>
using namespace std::
int main() {
    cout << "helloworld";
}
```

(D)
```
#include <iostream>
using namespace std
int main() {
    cout << "helloworld";
}
```

[1] In the previous question, what happens if you try anything but the correct choice? (E) The compiler errs (F) Time limit exists (G) Time limit exceeds (H) The program outputs `hollywood`

[2] `cout << int('a');` **outputs 97. What about** `cout << int('n') << int('t') << int('u');`**? For problems without choices, like this one, write your answer directly on the answer sheet.**

[3] Select all that do this correctly: take two numbers and output their sum? (At least one is correct.) (I)
```
int main () {;
    cout << cin() + cin();
}
```

(J)
```
int main () {
    cin >> (float A, float B);
    cout << A + B;
}
```

(K)
```
int main () {
    cin >> (float)A >> (float)B;
    cout << A + B;
}
```

(L)
```
int main () {
    float A, B;
    A >> B >> cin;
    cout << A + B;
}
```

(M)
```
int main () {
    float A, B;
    cin >> A, B;
    cout << A + B;
}
```

(N)
```
int main () {
    float A, B;
    cin >> A >> B;
    A += B;
    cout << B;
}
```

(O)
```
int main () {
    float A, B;
    cin >> A >> B;
    float B += A;
    cout << B;
}
```

(P)
```
int main () {
    float A, B;
    cin >> A >> B;
    A = B + A;
    B = A + B;
    cout << A;
}
```

[4] What does `1 + 2 * 3` evaluate to?

[5] What does `1 ^ 2 ^ 3 ^ 4 ^ 5 ^ 6 ^ 7 ^ 8 ^ 9 ^ 10 ^ 11 ^ 13 ^ 14 ^ 15 ^ 16` evaluate to?

[6] What does `33 + +44` evaluate to? (If there is something wrong, write down "error".)

[7] What does `10 / 2 (1 + 4)` evaluate to? (If there is something wrong, write down "error".)

[8] What does `10 / 23` evaluate to in the expression `float date = 10 / 23;`? (If there is something wrong, write down "error"; otherwise, I want two decimal places.)

[9] What does `20 // 24` evaluate to in the expression `float year = 20 // 24;`? (If there is something wrong, write down "error"; otherwise, I want four decimal places.)

[10] What is the most readable yet correct way to assign 4 percent of 25 to neo? (Q) `float neo = 4% * 25;` (R) `float neo = 25% * 4;` (S) `float neo = 0.05 * 25;` (T) `float neo = 4 / 100. * 25;`

[11] ***Yoda notation*** **is sometimes preferred because** (U) using underscores looks ugly (V) programmers sometimes mistype == as = (W) It is compatible with the Polish notation (X) It is compatible with the

[12] After the two lines to the right is executed, what is `a + b + c`?

```
int a = -2, b = 5, c = 2;
a += b; b -= c; c *= a;
```

[13] After two more lines being executed, what is `a + b + c`?

```
a /= b; b %= c; c != a;
a >= b; b <= c; c == a;
```

[14] After two more lines, what is `a + b + c`?

```
a >>= b; b <<= c; c = a;
a |= b; b ^= c; c &= a;
```

[15] To the right are some floating-point numbers and their binary representations. What is the binary representation of `64`? 0^n stands for n zeros. (Y) 0100010000000^{20} (Z) 0100001010000^{20} (A) None of the above

```
10111111100000000000000000000000 = -1
00111110000000000000000000000000 = 0.125
00111110100000000000000000000000 = 0.25
00111111000000000000000000000000 = 0.5
00111111100000000000000000000000 = 1
01000000000000000000000000000000 = 2
01000000010000000000000000000000 = 3
01000000100000000000000000000000 = 4
01000000101000000000000000000000 = 5
01000000110000000000000000000000 = 6
01000000111000000000000000000000 = 7
01000001000000000000000000000000 = 8
01000001100000000000000000000000 = 16
01000010000000000000000000000000 = 32
```

[16] What is the binary representation of `1.5`?

[17] What is the binary representation of `1.75`? (B) 0100000011100^{20} (C) 0100000001100^{20} (D) 0011111111100^{20} (E) None of the above

[18] What is the binary representation of `-2`?

[19] What is the binary representation of `-9`? (F) 1100000100000^{20} (G) 1100000100010^{20} (H) 1100000100100^{20}

[20] What is the binary representation of `1.0/3.0`?

[21] Which of the following is the most common way to define constant `PI` to the precision of 32-bit `float`? (I) `float PI = 3.14;` (J) `float PI = 3.1415927;` (K) `float PI = 3.1415926535 8979323846;` (L) `float PI = 3. 1415926535 8979323846 2643383279 5028841971 6939937510 5820974944 5923078164 0628620899 8628034825 3421170679;`

[22] The expression `0.4 * 0.5 == 0.02` evaluate to false because (M) they underflow as integers (N) they are not mathematically equal (O) `0.4` cannot be exactly represented in binary (P) `0.5` cannot be exactly represented in binary (Q) the compiler optimizes the expression incorrectly.

[23] Which two of the following are correct about compiler optimization flag `-Ofast`? (R) It compresses the source code (S) It makes the compiler run faster (T) It makes the program run faster (U) It might abuse undefined behavior toward its goal

[24] Select all the functions that are not commented out. (V) `StartTimer` (W) `CheckInternetAviliability` (X) `ConnectToServer` (Y) `LoadUserConfiguration` (Z) `MineBitcoinAndSendToNorthKorea` (A) `UpdateRemoteData` (B) `EndTimer`

```
// StartTimer();
   CheckInternetAviliability();
// ConnectToServer();
LoadUserConfiguration();
/*
 *  MineBitcoinAndSendToNorthKorea();
    UpdateRemoteData();
 */
EndTimer();
```

[25] In this paper-based midterm, if someone answers all the questions correctly, what would be the score?

[26] `int p2i = 10; int *Int = &p2i; *Int *= *Int; cout << p2i;` outputs (C) `10` (D) `20` (E) `100` (F) nothing, as it does not compile

[27] What is the purpose of the function to the right? (G) To sort an array (H) To rotate an array (I) To reverse an array (J) To shuffle an array (K) To summarize an array

```
void BreadthOfTheField(int A[], int a, int b) {
    while (a < b) {
        int t = A[a]; A[a] = A[b - 1]; A[b - 1] = t;
        BreadthOfTheField(A, a + 1, b - 1);
        break;
    }
}
```

[28] Consider another function as show to the right. I want to do that, but without using extra memory (i.e., the second array `B`). So, instead, I call `BreadthOfTheField(A, a, b)` with parameters `a` and `b` summing to?

[29] I then call `BreadthOfTheField(A, d, e)` with parameters `d` and `e` summing to?

```
void TrickOfTheWorld(int A[100], int c) {
    int B[100];
    for (int i = 0; i < 100; i++) {
        B[i] = A[(i + c) % 100];
    }
    for (int i = 0; i < 100; i++) {
        A[i] = B[i];
    }
}
```

[30] Finally I call `BreadthOfTheField(A, f, g)` with parameters `f` and `g` summing to?

[31] Simplify `a ^ b ^ c ^ d ^ e ^ f ^ g`

[32] What is the range of `signed int` if the compiler uses 32-bit? (L) 0 to 2^{31} (M) 0 to 2^{32} (N) 0 to $2^{32}-1$ (O) -2^{31} to 2^{31} (P) -2^{31} to $2^{31}-1$

[33] `uint16_t` is a type that represents an unsigned integer using 16 bits. Given that F_n, the nth Fibonacci number, is about $2^{0.69424n-1.161}$, select all that cannot be represented by `uint16_t`. (Q) F_{10} (R) F_{20} (S) F_{50} (T) F_{100} (U) F_{200} (V) F_{500}

[34] Define a generalization of Fibonacci sequence as $G_n = G_{n-1} + 2G_{n-2}$ with initial terms $G_0 = 1$ and $G_1 = 2$. Select all that cannot be represented by `uint32_t`. (W) G_{10} (X) G_{20} (Y) G_{50} (Z) G_{100} (A) G_{200} (B) G_{500}

[35] Define another generalization as $H_n = H_{n-1} + 3H_{n-2}$ with initial terms $H_1 = 1$ and $H_2 = 2$. Which three of the following cannot be represented by `uint64_t`? (C) H_{10} (D) H_{20} (E) H_{50} (F) H_{100} (G) H_{200} (H) H_{500}

[36] The *Year 2038 problem* refers to the overflow of which data type, given that the Unix time stores how many seconds has passed since 1970/01/01 00:00:00? (I) 8-bit integer (J) 16-bit integer (K) 32-bit integer (L) 64-bit integer

[37] After the loop to the right, with high probability, `S` is closest to (M) 300 (N) 3000 (O) 30000 (P) 300000 (Q) 99970000

```
int S = 0;
for (int i = 0; i < 100000000; i++) {
    if (rand() < 0.003) { S++; }
}
```

[38] Recall that modern CPUs have frequency ranging from 1 to 5 GHz. How long does it take to run the loop in the previous question? (Assuming no flags like `-Ofast`) (R) about 1–10 microseconds (S) about 1–10 seconds (T) about 1–10 days (U) about 1–10 years

[39] Which one finds the maximum element in `float A[100];`? (V)

```
float max1 = -INFINITY;
for(i = 0; i < 100; i++) {
    max1 |= A[i];
}
```

(W)
```
float max1 = 0;
for(i = 0; i < 100; i++) {
    if(A[i] > max1) {max1 = A[i];}
}
```

(X)
```
float min1 = A[0];
for(i = 1; i < 100; i++) {
    if(A[i] > min1) {min1 = A[i];}
}
```

[40] Which implementation is the fastest? (Do not worry about the correctness of the code.)

(Y)
```
bool IsPrime(int n) {
    /* boundary checks here */
    for (int i = 2; i < n; i++) {
        if (n % i == 0) return false;
    }
    return true;
}
```

(Z)
```
bool IsPrime(int n) {
    /* boundary checks here */
    for (int i = 2; i < n; i++) {
        if (n % i == 0) return false;
        if (i*i*i > n*n) return true;
    }
}
```

(A)
```
bool IsPrime(int n) {
    /* boundary checks here */
    for (int i = 2; i < n; i++) {
        if (IsPrime(i) && n % i == 0)
            return false;
    }
    return true;
}
```

(B)
```
bool IsPrime(int n) {
    /* boundary checks here */
    for (int i = 2; i < n; i++) {
        if (IsPrime(i) && n % i == 0)
            return false;
        if (i*i > n) return true;
    }
}
```

[41] Select the three scenarios where your function needs to take another function as an argument. (C) To sort an array (D) To reverse an array (E) To time a function (F) To remove nans from an array (G) To define a recursive function (H) To cache the values of a function

[42] Consider the program to the right. You want to make it as hard as possible to a beginner, but you also want to ensure that you yourself can always win. What should you set `attempts` to? (I) 1 (J) 2 (K) 3 (L) 4 (M) 5 (N) 6 (O) 7 (P) 8 (Q) 9 (R) 10

```
int attempts = ???;
int guess = 0;
int target = rand() % 100;
while (attempt > 0) {
    attempt -= 1;
    cin >> guess;
    if (guess > target) { cout << "Too high"; }
    else if (guess < target) { cout << "Too low"; }
    else { cout << "Win"; return; }
}
cout << "Lose: no more attempts";
```

[43] Two scenarios in which assertion can be used are (S) trying exceptions (T) catching exceptions (U) throwing exceptions (V) unit-testing a function (W) optimizing a function for speed

[44] Which of the following computes the factorial of `n`? (X)
```
fact = 1;
for (i = 1; i < n; i++) {
    fact *= i;
}
cout << fact;
```

(Y)
```
i = 1;
for (fact = 1; i <= n; i++) {
    fact *= n;
}
cout << fact;
```

(Z)
```
fact = i = 0;
for (i = 1; 1 < n; n--) {
    i *= n;
}
cout << i;
```

[45] Using `a`, `b`, `c`, `d`, `x`, `add(,)`, and `mul(,)`, instead of `+` and `*`, How to express cubic polynomial $ax^3 + bx^2 + cx + d$?

[46] For the previous question, can you use `mul` at most 5 times?

[47] The body mass index (BMI) is the quotient of weight in kilograms and height in meters squared. Complete the line that computes BMI.

[48] Complete the line that outputs `underweight` if the BMI is 17 or lower.

[49] Complete the line that outputs `overweight` if the BMI is 31 or higher.

```
float height, weight, BMI;
cout << "How tall are you in centimeters?";
cin >> height;
cout << "How heavy are you in kilograms?";
cin >> weight;
BMI = /* your code here */;
cout << "Your BMI is " << BMI;
if /* your code here */ ;
if /* your code here */ ;
if /* your code here */ ;
```

[50] Complete the line that outputs `military!` for BMI between 17 and 31.

[51] Select the correct structure regarding exception handling. (A) `catch { throw } try` (B) `catch { try } throw` (C) `throw { catch } try` (D) `throw { try } catch` (E) `try { catch } throw` (F) `try { throw } catch`

[52] What is the correct way to declare two pointers, both pointing to integers? (G) `int* a, b;` (H) `int *a, *b;` (I) `&int a, b;` (J) `int &a, &b;`

[53] Given `struct Circle { float x, y, r; };`, Write a function `struct Circle Scale(struct Circle a, float s) { /***/ }` that scales the circle by a factor of `s` while fixing the center.

[54] Given `struct Rectangle { float x1, y1, x2, y2; };`, Write a function `struct Rectangle Scale(struct Rectangle a, float s) { /***/ }` that scales the Rectangle by a factor of `s` while fixing the center. Here, `(x1, y1)` is the coordinate of the southwest corner, and `(x2, y2)` is the coordinate of the northeast corner.

[55] The function `Scale` is pass by? (K) Pass by value (L) Pass by pointer (M) Pass by reference (N) Pass by pointer value

[56] There are two DNA sequences `char A[200], B[200];`. Knowing that they came from a very old, therefore noisy, machine, how do I check if the second half of `A` matches the first half of `B`? (O) find the longest common substring of `A` and `B` (P) find the shortest common supersequence of `A` and `B` (Q) find the longest palindromic substring of `A[100..200] + B[0..100]` (R) compute the Hamming/Levenshtein distance between `A[100..200]` and `B[0..100]`

[57] The way the course slides hyper-link to each other is like a (S) stack (T) queue (U) hash table (V) binary tree (W) linked list (X) directed graph (Y) undirected graph (Z) double-ended queue (A) doubly-linked list (B) skipped linked list

[58] Among the choices of the previous question, which two data structures can be seen as special cases of the third one?

[59] Which one is the third data structure in the previous question?

[60] Reorder the following lines to make sense. (C) `tail->next = new Node; tail = tail->next; tail->data = count++;` `tail->next = NULL;` (D) `Node *head = new Node; head->data = count++; head->next = NULL;` (E) `struct Node { int data; Node *next; };` (F) `Node *tail = head;` (G) `int count = 0;`

[61] What is the value of `head->data` after the lines are executed?

[62] What is the value of `tail->data` after the lines are executed?

[63] In quantum information theory, a qubit is represented by a density matrix

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

where $q^\dagger$ is the conjugate of q. When ρ undergoes a not gate, it becomes

$$\text{NOT}(\rho) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} p & q \\ q^\dagger & r \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

With `struct Cmplx { float r, i; };`, and `struct Qubit { Cmplx p, q, r; };`, implement `struct Qubit Not(struct Qubit a) { /***/ }`. You may assume that we overload the operators so that complex numbers are added by `+` and multiplied by `*`.

[64] Currently, the largest prime known to human is $2^{136,279,841} - 1$. For the purpose of determining if a number of the form $M_p \coloneqq 2^p - 1$, where p is a prime, is a prime, GIMPS writes,

The programs that GIMPS users run perform a Fermat probable prime test. A successful test almost certainly is a new prime number. Once the GIMPS server is notified of a probable prime, several definitive Lucas–Lehmer primality tests are run using different programs on different hardware.

$2^{136,279,841} - 1$ passed the probable test on October 11. It then passed the definitive test on October 12 and 19, by different programs ran on different hardwares. On October 22, the media started reporting on this.

To check if M_p is a prime, the Fermat probable prime test checks $a^{M_p - 1} \equiv 1 \pmod{M_p}$ for some a. Lucas–Lehmer, on the other hand, checks if $s_{p-2} \equiv 0 \pmod{M_p}$, where $s_0 \coloneqq 4$ and $s_i \coloneqq s_{i-1}^2 - 2$. Now, fix an a, say 2, what is the time complexity of Fermat probable prime test in terms of p? Note that multiplying two n-bit integers costs complexity $O(n \log n)$, and modulo costs the same.

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, $1 \leq L \leq 10^{16}$, 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.

```
#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`