1 Classes and Structures

```
class Rabbit {
private:
    int age;
    int health;
public:
    void next_day() { age += 1; }
    bool too_old() {
        // return ???;
    }
// TODO: get_health
// TODO: set_health
}
```

[1] From the unfinished homework to the right, we can infer what? Select all that apply. (A) `age` is a private member (B) `too_old` is a public method (C) `next_day` is a public method (D) `health` is a private variable (E) `Reddit` is the name of the class

[2] `too_old` should return (F) age (G) `age < 36` (H) `age > 36` (I) `age = 36` (J) `age == 36`

[3] Write a getter for `health`.

[4] Write a setter for `health`, but make it zero if the input is negative.

```
class IntMatrix {
public:
    int size;
    int *entries;
    IntMatrix(int s) : size(s) {
        entries = new int[size * size];
    }
    ~IntMatrix() {
        delete[] entries;
    }
// TODO: gentry
// TODO: sentry
}
```

[5] Here is another unfinished homework. Should I write `IntMatrix A;`, without specifying `size`? (K) Yes, because we can always change `size` later. (L) Yes, it is a good way to initialize a blank `A`. (M) No, without `size` it is unclear how much memory to allocate. (N) No, because `size` is a private member and cannot be accessed from outside.

[6] Is `IntMatrix` a POD (Plain Old Data)? (O) No but it would be if size is a constant. (P) No because it uses dynamic memory allocation. (Q) Yes because it does not inherit from a base class. (R) Yes because it is made of integers and pointers, which are PODs.

[7] Write a 2-parameter getter for `entries`. You can assume that the inputs are always valid.

[8] Write a 3-parameter setter for `entries`. You can assume that the inputs are always valid.

[9] Suppose I have a 3x3 matrix `A` and I copy this matrix by `IntMatrix; B = A`, what could go wrong? (S) `B` will use twice as much memory as `A` (T) Modifying `B[5]` will also modify `A[5]` (U) `B` does not have any content; `B[0]` is undefined (V) `B` will be a deep copy of `A` and slows down the program.

[10] Overload `IntMatrix operator=(const IntMatrix& A)` to overcome the issue in the previous problem.

[11] What is the main difference between `struct` and `class`? (W) `class` is for data (X) `struct` is for functions (Y) `class` is public by default (Z) `struct` is public by default

2 Overloading

[12] Which one is correct about `operator+` (A) there are binary + and ternary + (B) you cannot overload + for exceptions (C) you can overload + for istream and ostream (D) your definition has to make `a + b` equivalent to `b + a` (E) your definition has to make `(a + b) + c` equivalent to `a + (b + c)`

[13] Overload `string operator*(string s, int n)` to return `s` repeated `n` times.

[14] Overload `string operator/(string s, int n)` to return the first `1/n` portion of the string. (Round up to the nearest whole character, so `ntu/2 = nt`).

[15] Overload `string operator*(string s, string t)` in a way such that `(a * b) * c` is not the same as `a * (b * c)`.

[16] Why, sometimes, it is `complex operator*(complex a)` instead of a and b? (F) It is in the class definition (G) It is implementing complex conjugate (H) It is overloading the dereferencing operator

[17] I hope that `Complexity C1 = 3; cout << C1 << endl;` will print the string `O(n^3)` to the screen.

```
class Complexity {
public:
    int degree;
    Complexity(int d) : degree(d) {}
    friend ostream& operator<<(ostream& oyster, const Complexity& x1) {
        // write what is missing on the answer sheet
        return oyster;
    }
};
```

[18] Overload addition so that `O(n^1) + O(n^2) + O(n^3)` results in `O(n^3)`.

```
Complexity operator+(const Complexity& LHS, const Complexity& RHS) {
    // write what is missing on the answer sheet
}
```

[19] Overload multiplication.

```
Complexity operator*(const Complexity& here, const Complexity& there) {
    // write what is missing on the answer sheet
}
```

3 Polymorphism

[20] From the code to the right, `Rabbit R1; R1.eat();` results in (I) health of R1 increased by 0 (J) health of R1 increased by 1 (K) health of R1 increased by 2 (L) health of R1 increased by doubled

[21] `Animal *A2 = &R1; A2->eat()` results in (M) health of R1 increased by 0 (N) health of R1 increased by 1 (O) health of R1 increased by 2 (P) health of R1 doubled

[22] `Bear B3; B3.eat(*A2)` results in (Q) health of R1 increased by 0 (R) health of B3 increased by 1 (S) health of R1 increased by 2 (T) health of B3 increased by R1.health

[23] Which two of the following statement about polymorphism are true? (U) Polymorphism requires all derived classes to implement the methods again. (V) Polymorphism means defining a function multiple times with different signatures (W) Polymorphism allows objects of different classes to be treated as objects of a common base class (X) Polymorphism is about manipulating objects by what they can do, instead of what they actually are

```
class Animal {
 public:
    int health;
    virtual void eat() {
        health += 1;
    };
};
class Rabbit : public Animal {
public:
    void eat() {
        health += 2;
    };
};
class Bear : public Animal {
public:
    void eat() {
    }
    void eat(Rabbit R1) {
        health += R1.health;
        R1.health = 0;
    };
};
```

[24] Given that A4 and A5 are pointers to Animals, which could have been bears or rabbits. How would you determine if A4 can eat A5? Specify which member/method you want to modify.

```
// Example answer
I want to modify Animal by adding a function called ...
Then, I want to modify Bear by ...
```

[25] Show me an example of "if A4 can eat A5, let it eat".

4 Lambda

```
auto f = [](auto x) { return 3*x + 1; }
auto g = [](auto x) { return x/2; }
int y = 1;
auto j = [=y](auto x) { return x*y; }
auto k = [&y](auto x) { return x*y; }
y = 2;
```

[26] What is the result of `f(3.1)`?

[27] What is the result of `f(g(3.14))`?

[28] What is the result of `f(g(j(3.141)))`?

[29] What is the result of `f(g(g(k(3.1415))))`?

[30] Church numeral one is defined as `[](auto f) { return [=](auto x) { return f(x); }; }`. Church numeral two is defined as `[](auto f) { return [=](auto x) { return f(f(x)); }; }`. What do you think Church numeral three is?

[31] If `C` is the Church numeral for some integer `c`, what is the Church numeral for `c + 1`?

[32] If `A` is the Church numeral for some integer `a` and `B` is the Church numeral for some integer `b`, what is the Church numeral for `a + b`?

[33] What is the Church numeral for `a * b`?

5 Template

[34] `From0to100` is a `vector<int>` containing 11 elements from 0 to 10. What is the result of `accumulate(From0to100.begin(), From0to100.end(), 1, [](int a, int b) { return a * b; });`?

[35] What is the result of `accumulate(From0to100.begin(), From0to100.end(), 1, [](int a, int b) { return a + b; });`?

[36] What is the result of `accumulate(From0to100.begin(), From0to100.end(), 1, [](int a, int b) { return a * b + a + b; });`?

[37] `ScoreVect` is a `vector<int>` containing the scores of each student in a class. What data structure cannot help me understand the distribution of scores?

(Y)
```
set<int> ScoreDist;
for (int s : ScoreVect) {
    ScoreDist.insert(s);
}
```

(Z)
```
multiset<int> ScoreDist;
for (int s : ScoreVect) {
    ScoreDist.insert(s);
}
```

(A)
```
map<int, int> ScoreDist;
for (int s : ScoreVect) {
    ScoreDist[s] += 1;
}
```

(B)
```
unordered_map<int, int> ScoreDist;
for (int s : ScoreVect) {
    ScoreDist[s] += 1;
}
```

[38] Rank the other three data structures from most handy to least handy for understanding the distribution.

[39] I want to sort `tuple<char, int, double>` reverse co-lexicographically. An example for `tuple <char, int>` is `(b, 2)`, `(a, 2)`, `(b, 1)`, `(a, 1)`. Note that the fourth argument of `sort` is a lambda function $f(a, b)$ determining if a should come before b.

```
vector<pair<int, float>> vee = {{1, 3.14}, {2, 2.72}, {3, 1.41}, ...};
sort(vee.begin(), vee.end(), [](const auto& a, const auto& b) {
    // write what is missing on the answer sheet
});
```

[40] I want to be able to add two queues of type `queue<Car>` that represents the "fair" way when two lanes of cars merge into one lane. That is, the first car from the left lane goes first, the first car from

the right lane goes next, then the second car from the left lane, then the second car from the right lane, and so on.

[41] Write `TyMatrix`, which is a generalization of `IntMatrix` to arbitrary entry type `Ty`. (Must include: constructor, destructor, copy constructor, setter, getter)

6 Threads

[42] A thread is (C) a process (D) a social media (E) a physical CPU core (F) a basic unit of execution within a process, a sequence of instructions that can run independently, often in parallel with other threads.

[43] Which one is correct? (G) More threads always mean faster execution (H) Fewer threads always mean faster execution (I) The number of threads should be determined by benchmarking (J) The number of threads should always equal to the number of CPU cores

[44] Which of the following for-loop benefit the most from parallelization?

(K)
```
for (int i = 0; i < 10000; i++) {
     A[i] = A[i-1] + A[i-1];
}
```

(L)
```
for (int i = 0; i < 10000; i++) {
    B[i] = A[i-1] + A[i-1];
    A[i] = B[i];
}
```

(M)
```
for (int i = 0; i < 10000; i++) {
     A[i + 1] = log(A[i] + 100);
}
```

(N)
```
for (int i = 0; i < 10000; i++) {
     A[i] *= atan2(A[i], A[i + 1]);
}
```

[45] I want to make evaluate `f(A[i])` and put the result in `B[i]` for some complicated function `f`. Rank the following options from fast to slow. Note that how much time `f(x)` costs depends on `x`. (O) Give `size/2` threads two elements each (P) Use one thread to handle `A[2*i]` and another thread to handle `A[2*i+1]` (Q) Use one thread to handle `A[0...size/2]` and another thread to handle `A[size/2...size]` (R) Give each of the two threads `sqrt(size)` elements and give it more when it finishes

[46] By default, what happens if the main thread and a subordinate thread try to read to the same memory location? (S) Compile error (T) Immediate segfault (U) Nothing special will happen; both will read (V) The memory is locked but it is unclear which thread will wait (W) The memory is locked and the subordinate thread waits the main thread

7 Optimization

[47] Why is it important to time your programs? (X) To compare different algorithms (Y) To identify performance bottlenecks (Z) To ensure it runs within acceptable limits (A) To avoid wasting time on premature optimization (B) All of the above are correct

[48] What is not the reason that timing a program multiple times results in less and less time? (C) The CPU is getting warmer (D) The CPU turns on turbo mode (E) The CPU predicts branches better

[49] Which of the following is the proper way to time a function that takes a few nanoseconds to run? (`Now()` is `std::chrono::high_resolution_clock::now();`)

(F)
```
auto start = Now();
function_to_time();
auto end = Now();
cout << end - start;
```

(G)
```
auto start = Now();
function_to_time();
auto d1 = Now() - start;
auto d2 = duration_cast<seconds>(Now() - start);
cout << d2.count();
```

(H)
```
auto start = Now();
for(int i = 0; i < 10000; i++)
    function_to_time();
auto end = Now();
cout << Now() - start;
```

(I)
```
auto start = Now();
for(int i = 0; i < 10000; i++)
    function_to_time();
cout << duration_cast<nanoseconds>
        (Now() - start).count();
```

[50] Should I write `a << 1` whenever I mean `a * 2`? (J) No because `a + a` is faster (K) Yes because bit-shifting is faster (L) No because `a + a` is more readable (M) Yes because bit-shifting is more readable (N) No because we should leave optimization to the compiler

[51] Why `for(auto i : zoo)` is preferred over the traditional loop? Select the wrong answer. (O) untyped `i` runs faster (P) It avoids off-by-one errors (Q) It avoids writing `i` three times (R) It expresses the intent more clearly (S) It leaves the optimization to the compiler

[52] Rank the memories from fast to slow (T) cloud storage (U) HDD (V) L1 (W) L2 (X) L3 (Y) RAM (Z) register (A) SSD

[53] Rank the memories from more storage to less storage

8 Culture and Misc

[54] What is wrong with the code to the right? (B) It does not check if it overwrites an existing file (C) It does not check if the file is closed successfully (D) It does not check if the file is opened successfully

```
ofstream fout("output.md");
fout << "## Hello, world!";
fout.close();
```

[55] C++ Core Guidelines advises writing like this because (choose all that apply) (E) it specifies the file name at the beginning (F) it lets other header files to test if this one is included (G) it includes the header file only if it has not been included (H) it tests if the compiler is capable of conditional compilation (I) it includes a header file if it is compatible with the compiler

```
// file foobar.h:
#ifndef LIBRARY_FOOBAR_H
#define LIBRARY_FOOBAR_H
// ... declarations ...
#endif // LIBRARY_FOOBAR_H
```

[56] Google C++ Style Guide is against exceptions because (choose all that apply) (J) the company is made of exceptional engineers (K) catching all types of exceptions can be tedious (L) an exception can penetrate multiple layers of function calls (M) most modern languages are already equipped with exceptions so C++ does not have to

[57] In a function definition, local static variable is preferred over global variable (N) to stop other functions from changing it (O) to allow this function to change it (P) to force allocation in registers instead of RAM (Q) to that ensure the variable is initialized only once

```
int random101(int min, int max) {
    static int seed = 12;
    seed = (seed * 12) % 101;
    return min + seed % (max - min + 1);
}
```

[58] In a function definition, local static variable is preferred over constant variable (R) to stop other functions from changing it (S) to allow this function to change it (T) to force allocation in registers instead of RAM (U) to that ensure the variable is initialized only once

[59] At the very beginning of the main function, `cout << random101(0, 10) + random101(0, 10) + random101(0, 10);` results in?

[60] What are on the call stack? (V) Templates and macros (W) Structures and classes (X) Overflows and underflows (Y) Static variables and dynamic variables (Z) Local variables and function parameters

[61] [62] [63] Read the instruction very carefully.

Write a template function `template<typename CC> void FourierTransform(int n, CC *A, CC *B)` that implements the Fourier transform of an array-like `A` of length `n`, defined as

$$B[k] := \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} A[j] \exp\Big(-\frac{2\pi ijk}{n}\Big)$$

where $k = 0, 1, \ldots, n-1$.

`CC` is a customized class for complex numbers. here are things you can do:

- ○ You can access the `j`th element of `A` by `A[j]`.
- ○ `CC() // default constructor returns 3.14159265358979323846 + 0i`
- ○ `CC operator+(const CC& a, const CC& b)`
- ○ `CC operator*(const CC& a, const CC& b)`
- ○ `CC operator/(const CC& a, const CC& b)`
- ○ `CC Cexp(CC a) // exp function for CC`
- ○ `CC Csqrt(CC a, int b) // returns sqrt(b)`

You do not have to worry about overflow, arithmetic error, and out-of-bound. I will use `C1` to test your code.

```
#include <iostream>
#include <cmath>
#include <ctime>
using namespace std;

class C1 {
private:
    double re, im;
public:
    C1() : re(3.14159265358979323846), im(0) {}
    C1(double r, double i) : re(r), im(i) {}
    C1 operator+(const C1& other) const {
        return C1(re + other.re, im + other.im);
    }
    C1 operator*(const C1& other) const {
        double r = re * other.re - im * other.im;
        double i = re * other.im + im * other.re;
        return C1(r, i);
    }
    C1 operator/(const C1& other) const {
        double denom = other.re * other.re + other.im * other.im;
        double r = (re * other.re + im * other.im) / denom;
        double i = (im * other.re - re * other.im) / denom;
        return C1(r, i);
    }
    friend ostream& operator<<(ostream& oyster, const C1& other) {
        oyster << other.re << " + " << other.im << "i";
        return oyster;
    }
    C1 Csqrt(int b) {
        if (b >= 0) {
            return C1(sqrt(b * 1.), 0.);
        }
        else {
            return C1(0., sqrt(-b * 1.));
        }
    }
    C1 Cexp(C1 a) {
        return C1(exp(re) * cos(im), exp(re) * sin(im));
    }
};
template<typename CC>
CC Csqrt(CC a, int b) { return a.Csqrt(b);}

template<typename CC>
CC Cexp(CC a) { return a.Cexp(a);}

template<typename CC>
void FourierTransform(int n, CC* A, CC* B, bool inverse=0) {
    // your code here
    // your code here
    // your code here
}

// Example usage:
int main() {
    // srand(time(NULL));

    int n = 5; // try n = 5, 10, 15, 20, 25;
    C1 A[n];
    C1 B[n];
    C1 C[n];
    for (int i = 0; i < n; i++) {
        A[i] = C1(1 + rand() % 9, 1 + rand() % 9);
        cout << "A" << i << " is " << A[i] << endl;
    }
    cout << endl;
    FourierTransform(n, A, B);
    FourierTransform(n, B, C);
    for (int i = n; i > 0; i--) {
        cout << "C" << i % n << " is " << C[i % n] << endl;
        // this should output the sequence as A
    }

    return 0;
}
```

[64] [65] [66] Now implement the inverse transform.

The inverse Fourier transform is defined as

$$B[k] := \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} A[j] \exp\left(\frac{2\pi ijk}{n}\right)$$

Change the signature of `FourierTransform` to `template<typename CC> void FourierTransform(int n, CC *A, CC *B, bool inverse=0)` so that `FourierTransform(n, A, B)` still means the forward transform, and `FourierTransform(n, A, B, 1)` means the inverse transform.

To test your code, use random `A` and check if `FourierTransform(n, A, B); FourierTransform(n, B, C, 1);` results in `C` being the same as `A`;

[67] [68] [69] [70] To get bonus points, you must obey the following rule.

- × You have to pretend that you have no idea how `CC` is implemented.
- × For example, do not access the real and imaginary parts directly.
- × Do not use the constructor `CC(double a, double b)`.

- × Do not use the initialization syntax `CC A{a, b};` and `CC A = {a, b}`.
- × You can only modify `FourierTransform`.

I will use both `C1` and `C2` to test your code. Do not assume that you know `C2`. The following is just a demonstration that `C2` is defined differently from `C1`. It is not the actual `C2` that will be used.

```
class C2 {
private:
    double im, re;
    // when I test your code for bonus points, the order can be
    // re, im
    // or im, re
    // or garbage, re, garbage, im, garbage.
    // Do not assume that you know the order of the variables.
public:
    C2() : re(3.14159265358979323846), im(0) {}
    C2(double r, double i) : re(r), im(i) {}
    C2 operator+(const C2& other) const {
        return C2(re + other.re, im + other.im);
    }
    C2 operator*(const C2& other) const {
        double r = re * other.re - im * other.im;
        double i = re * other.im + im * other.re;
        return C2(r, i);
    }
    C2 operator/(const C2& other) const {
        double denom = other.re * other.re + other.im * other.im;
        double r = (re * other.re + im * other.im) / denom;
        double i = (im * other.re - re * other.im) / denom;
        return C2(r, i);
    }
    friend ostream& operator<<(ostream& oyster, const C2& other) {
        oyster << other.re << " + " << other.im << "i";
        return oyster;
    }
    C2 Csqrt(int b) {
        if (b >= 0) {
            return C2(sqrt(b * 1.), 0.);
        }
        else {
            return C2(0., sqrt(-b * 1.));
        }
    }
    C2 Cexp(C2 a) {
        return C2(exp(re) * cos(im), exp(re) * sin(im));
    }
};
```

[71] [72] [73] The problem of linear regression is to find the best coefficients `double a, b;` such that `aX[j] + b` is very close to Y[j] for all `j`, where `double X[n], Y[n]` are arrays of the data you want to fit.

What closeness mean depends on your applications. In this problem, we want to minimize the regret function

$$R(a,b) = \sum_{j=0}^{n-1} \Big| aX[j] + b - Y[j] \Big|^{1.5}$$

Define `pair<double, double> LinearRegression(int n, double *X, double *Y)` that does the following.

- Initialize both a and b to zero.

- Iterate from `i = 1` to `1 = 10000`.
- Let `aup = a + 1/sqrt(i)` and `adown = a - 1/sqrt(i)`.
- if aup fits the data better, that is, if `R(aup, b) < R(adown, b)`, replace a by aup. Otherwise, replace a by adown.
- Now perturb b and replace b with the one that fits better.
- After the iteration, return `make_pair(a, b)`.

You can use the following main function to test your code.

```
int main() {
    // srand(time(NULL));

    int n = 100;
    double a = rand() % 11 - 5, b = rand() % 21 - 10;
    cout << "true (a, b) = ("<< a << ", " << b << ")" << endl;
    double X[n], Y[n];
    for (int i = 0; i < n; i++) {
        X[i] = i - n / 2;
        Y[i] = a * X[i] + b + (rand() % 3) - 1;
    }
    auto guess = LinearRegression(n, X, Y);
    cout << "guess = ("<< guess.first << ", " << guess.second << ")" << endl;
    return 0;
}
```

The error should be about 1%–5%;

[74] [75] [76] Consider the following main function

```
int main() {
    // srand(time(NULL));

    int n = 100;
    double a = rand() / double(RAND_MAX);
    double b = rand() / double(RAND_MAX);
    a = 10 + a * 10;
    b = 0.9 * a * b;
    double X[n], Y[n];
    for (int i = 0; i < n; i++) {
        X[i] = 10 * i / n
        Y[i] = sin(2 * pi * (a * X[i] + b))
    }
    auto guess = Guess(n, X, Y);
    cout << "true a - guess a is " << endl;
    cout << "true b - guess b is " << endl;
    return 0;
}
```

Do whatever you want to approximate a and b to within 10^{-5} error. But your code should finish in 1 second using the computers in the classroom. Please do not rely on the fact that `RAND_MAX` can be as small as 32767. I am going to make sure that all `double` in the range will be selected somewhat uniformly.

[77] [78] [79] [80] 10^{-10} **error.**

Computer Programming English Taught - Final Exam

考試時間 Test time 15:51:00 pm to 18:18:00 pm. 以教室投影機時鐘為準 Using the projected clock. 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Xi. 禁止外部資源 No external resources. 舉手問問題 Raise your hand to ask questions.

姓名 Name (zh or en) 學號 Student ID

[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] [80]