Midterm Exam of Programming Language II (CSE, NTOU) ID: _____

09:20 – 12:05, 11 April 2025; Room INS101 Name: _____

Note: Cell phones and any smart device or calculator are forbidden.

1. (5%) Please modify the definition of class `B` to make the program run successfully WITHOUT changing the body of `main()`.

```
#include<iostream>

class A {
public:
     int x = 0;
protected:
     int y = 1;
};

class B : private A {
// modification in the class body is required
};

int main (){ // the output: 1
     B obj1;
     std::cout << obj1.y << std::endl;
     return 0;
}
```

2. (5%) Please write down the output appearing on the screen.

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

class A {
public:
    A() { cout << "C1 "; }
    ~A() { cout << "D1 "; };
};
class B: public A {
public:
    B() { cout << "C2 "; }
    ~B() { cout << "D2 "; };
};
```

```
class C: public B {
public:
    C() { cout << "C3 "; }
    ~C() { cout << "D3 "; };
};
int main() {
     C obj;
     return 0;
}
```

3. (5%) Please write down the output appearing on the screen when the input is "`NTOU CSE`".

```
#include <iostream>

int main() {
     std::string str;
     std::cin >> str;
     std::cout << str << std::endl;
     return 0;
}
```

4. (5%) Please correct the lines in bold face so that `func()` will print "`Alan Bob Curry`" on the screen when user enters "`Alan Bob Curry`" on the console.

```
void func() {
    string zodiac[3];
    for (string s : zodiac)
        cin >> s;
    for (string s : zodiac)
        cout << s << " ";
}
```

5. (10%) Correct the following program (including adding appropriate **constructors**) to fit the sample output.

```
#include <iostream>

namespace NTOU {
    class Student{ // modify the class
    private:
        std::string name;
        int ID;
    };
}
namespace NYCU { // modify the class
    class Student {
    private:
        std::string name;
        int ID;
    };
}
```

```
using namespace NTOU;
using namespace NYCU;

int main() {
// correct the following two lines
    Student s1("Elvis", 100);
    Student s2("David", 200);
    return 0;
}
```

Sample output:

```
NTOU: Elvis, 100
NYCU: David, 200
```

6. (5%) Please predict the output the following program.

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

class Test {
    static int x;
public:
    int y = 0;
    void increase() { x++; y++; }
    void show() {
      cout << x << " " << y << endl;
    }
};
```

```
int Test::x = 1;

int main() {
    Test obj1, obj2;
    obj1.increase();
    obj1.show();
    obj2.increase();
    obj2.show();
}
```

7. (5%) Please correct the constructor in the following class definition.

```
class ConstRef {
public:
    ConstRef(int ii);
private:
    int i;
    const int ci;
    int &ri;
};

ConstRef::ConstRef(int ii) { i = ii;  ci = ii; ri = i ; }
```

8. (10%) Correct the move constructor and the first line in the main function body to activate the move constructor.

```
class Data {
public:
    Data() { std::cout << "Constructor of no parameter." << endl; }
    // Move Constructor
    Data(Data&) { std::cout << "Move constructor here!" << endl; }
};
Data createData() { return Data(); }
int main() {
    Data d = createData();  // Move constructor is supposed to be used here
    return 0;
}
```

9. (5%) Correct the following program and predict its output.

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

int main() {
     int array[3][3] = {{3, 4, 5}, {6, 7, 8}, {0, 1, 2} };
     for (&row in array) {
       for (entry in row) {
             if (entry % 2 == 0) cout << " 0 ";
             else cout << " 1 ";
       }
       cout << endl;
     }
     return 0;
}
```

10. (5%) Predict the output of following program (fix any error if necessary).

```
#include<iostream>
class A {
  int x;
public:
  A() { x = 0; }
  A(int i): x(i) { }
  void setX(int i) { x = i; }
  void print() { std::cout << x; }
};
class B: virtual public A {
public:
  B(): A(10) {  }
};
```

```
class C: virtual public A {
public:
  C(): A(20) {  }
};

class D: public B, public C { };

int main() {
     D d;
     d.print();
     return 0;
}
```

11. (10%) Please correct class `Person` to make it become a valid **abstract** class with a **destructor**.

```
class Person {
     int info;
public:
     virtual Person() = default;
     Person(int x) { std::cout << "Person(int) called" << std::endl; }
     PersonInfo() { std::cout << "Person's info" << std::endl; }
};
```

12. (5%) Please point out which line has an error in the program below.

```
class MyClass {                                                  // line 1
private:                                                         // line 2
    int myData = 0;                                              // line 3
public:                                                          // line 4
    MyClass() = default;                                         // line 5
    MyClass(int num): myData(num) {}                             // line 6
    int modMyData1() const { return ++myData; }                  // line 7
    int modMyData2() const { return myData + 1; }                // line 8
    int modMyData3()       { return ++myData; }                  // line 9
};                                                               // line 10
```

13. (10%) Please **point out which line has an error in the program below (line 1~4)** and **modify the structure `Derived`** to make the erroneous line be successfully compiled.

```
struct Base { int memfcn(); };
struct Derived : Base {  int memfcn(int);  };
Derived d;
Base b;
b.memfcn();                                                      // line 1
d.memfcn(10);                                                    // line 2
d.memfcn();                                                      // line 3
d.Base::memfcn();                                                // line 4
```

14. (15%) Please complete the following tasks:
    (1) (5%) Define an abstract class `Animal` with a pure virtual `makeSound()` and a virtual destructor.
    (2) (5%) Add a destructor for each derived class so that a message comes out once an object is destroyed.
    (3) (5%) Correct the `main()` function to make the following code be valid.

```
// define class Animal ...

class Dog : public Animal {
public:
    void makeSound() override { std::cout << "Dog: Woof!" << std::endl; }
};
class Cat : public Animal {
public:
    void makeSound() override { std::cout << "Cat: Meow!" << std::endl; }
};
int main() {
    Animal* zoo[2]; // get three pointers of animals
    zoo[0] = new Dog();
    zoo[1] = new Cat();
    for (v in zoo) { // correct this line if necessary
        v->makeSound();
    }
    for (v in zoo) { // correct this line if necessary
        delete v;
    }
    return 0;
}
```

Sample output:

```
Dog: Woof!
Cat: Meow!
Dog is dead.
Cat is dead.
```

15. (5%) Please modify the following program by freeing the dynamically allocated memory in the correct way.

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

int main()
{
    auto *pi = new int[5];
    for (int i=0; i<5; i++) cin >> pi[i];
    for (int i=0; i<5; i++) cout << pi[i] << endl;
    return 0;
}
```

16. (5%) Please predict the output of the following program.

```
auto p = make_shared<int>(17);
auto q(p);
auto r = make_shared<int>(23);
r = q;
auto s(r);
cout << r.use_count(); // what's the output?
```