

Student Number:

Name:

Write down answers in-between questions. Please answer using short sentences.

The back of each page can be used for practice, but DO NOT write down the answer on the back.

Be sure to write your student number and name on each page.

1. (12 pts) True / False questions:

- 1) Function overloading is the redefinition of base class function in its derived class. ()
- 2) Placing a member variable in the protected section of the base-class declaration allows the derived classes to access that member variable. ()
- 3) (For most compilers that use "vtable" mechanism for dynamic bounding,) An instance of any C++ class has a hidden member variable named "vptr" or similar. ()
- 4) A vptr of a class instance points to the vtable of its declared type. ()
- 5) If a function t() is declared as a friend function in the class A, t() can directly access private variables in the class A. ()
- 6) If "friend class O" is declared in the class A, member functions in the class A can access all the member functions and variables in the class O. ()
- 7) Static member functions cannot be virtual. ()
- 8) Static member variables can only be accessed by static member functions. ()
- 9) Template is a feature of C++ that allows us to write one code for different data types. ()
- 10) Template is an example of compile time polymorphism. ()
- 11) A catch handler cannot access local objects defined within its try block. ()
- 12) A catch(...) should always be placed first in the list of handlers following a try block. ()

2. (6 pts) In general, an operator whose result is a new object or the copy of "this" object returns the new or copied object by value, and whose result is an existing but modified "this" object returns the reference to the modified "this" object, although the C++ language rarely puts constraints on operator overloading. Fill in blanks (a)-(f) with the proper return type of the operators based on this policy, to make them behave as similar as possible to the built-in operators.

```
class A {  
  __ (a) __ operator =(const A& a);  
  __ (b) __ operator +(const A& a) const;  
  __ (c) __ operator -() const;  
  __ (d) __ operator +=(const A& a);  
  __ (e) __ operator ++();  
  __ (f) __ operator ++(int);  
};
```

Student Number:

Name:

3. (4 pts) Write down the expected output of the following programs. The position of newline in the answer must also be correct.

```
#include <iostream>
using namespace std;
class Parent {
public:
    Parent() { cout << " Parent"; }
    ~Parent() { cout << " ~Parent"; }
};
class Child : public Parent {
public:
    Child() { cout << " Child"; }
    ~Child() { cout << " ~Child"; }
};
class Test : public Child {
public:
    Test() { cout << " Test"; }
    ~Test() { cout << " ~Test"; }
};
int main() {
    {
        Child child;
        cout << endl;
    }
    cout << endl;
    {
        Test test;
        cout << endl;
    }
    cout << endl;
    return 0;
}
```

4. (5 pts) Select the expected output and result for the following program.

```
#include<iostream>
#include<string>
using namespace std;
void f() {
    throw string("error occurred");
    return;
}
int main(){
    try{
        f();
    }
    catch(const char* e)
    {
        cout << "caught an err!\n";
    }
    return 1;
}
```

- | | |
|---------------------------------|---|
| 1) "caught an err!" is printed. | 2) "error occurred" is printed. |
| 3) A compile error occurs. | 4) Program termination due to uncaught exception. |

Student Number:

Name:

5. (7 pts) When compiling the following C++ program, indicate whether each command line will compile correctly or if an error will occur by marking (putting a circle around) 'Okay' or 'Error'.

```
class ClassA
{
public:
    int a_var;
    void FunctionA(int i, double j);
    double& b_var()          { return b_var_; }

private:
    static double b_var_;
    void FunctionB(int i, double j);
};

int main ()
{
    ClassA c;
    const ClassA& rc = c;           // Okay / Error
    c.a_var = 10;                   // Okay / Error
    c.b_var() = 1.0;                // Okay / Error
    c.a_var = rc.b_var();           // Okay / Error
    double val = c.a_var + c.b_var_; // Okay / Error
    rc.FunctionA(c.a_var, 1.0);     // Okay / Error
    c.FunctionB(c.a_var, 2.0);      // Okay / Error
    return 0;
}
```

6. (6 pts) List all the functions whose addresses are in the vtable of the Student class.

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

class Person
{
public:
    virtual void talk();
    virtual void getName();
    void sayHi();
};

class Student : public Person
{
public:
    virtual void talk();
    virtual void study();
};

class CSStudent : public Student
{
public:
    virtual void talk();
    virtual void writeCode();
};
```

Student Number:

Name:

7. (6 pts) Add or modify just one line in the following code to compile successfully and print the output below. Only one line modification or one line addition is allowed; combinations such as one line modification and one line addition, or any other variations are not allowed. You cannot change the name of functions, variables, classes, and class inheritance relationships. You cannot change the bodies of A::getMember() and B::getMember() as well.

```
1. #include <iostream>
2. using namespace std;
3.
4. class A
5. {
6. protected:
7.     int memberA;
8. public:
9.     A(int n):memberA(n) { }
10.    int getMember() { return memberA; }
11. };
12.
13. class B: public A
14. {
15. public:
16.     B() {}
17.     int getMember() { return memberA; }
18. };
19.
20. int main()
21. {
22.     A a(1);
23.     cout << a.getMember() << endl;
24.
25.     B b;
26.     cout << b.getMember() << endl;
27.
28.     return 0;
29. }
```

Output:

1
100

8. (6 pts) Modify the myMax function template so that the following outputs are displayed when the program is executed. Preserve myMax as a function template without converting it into a regular function. Do not modify the main function and do not modify the return type of the myMax function template.

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

template <typename T>
T myMax(T x, T y)
{
    return (x > y)? x: y;
}

int main()
{
    cout << myMax(3.3, 3) << endl;
    cout << myMax(3, 3.3) << endl;

    return 0;
}
```

Output:

3.3
3

Student Number:

Name:

9. (5 pts) What is the expected output of this program? (If an error is expected, just write down "error").

```
#include <iostream>
using namespace std;
class Person {
public:
    ~Person()    {cout << "a ";}
};

class Student : public Person {
public:
    virtual ~Student()    {cout << "b ";}
};

class CSStudent : public Student {
public:
    ~CSStudent()    {cout << "c ";}
};

class Faculty : public Person {
public:
    ~Faculty()    {cout << "d ";}
};

int main() {
    Person* p1 = new Faculty;
    Person* p2 = new CSStudent;
    delete p2;
    delete p1;
}
```

10. (5 pts) What is the expected output of this program? (If an error is expected, just write down "error").

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

class Animal
{
public:
    virtual string talk() { return "..."; }
    virtual int getNumLegs() { return 0; }
};

class Cat : public Animal
{
public:
    virtual string talk() { return "Meow!"; }
    virtual int getNumLegs() { return 4; }
};

int main() {
    Animal* pa = new Animal;
    Cat* pd = dynamic_cast<Cat*>(pa);
    if(pd)
        cout << pd->getNumLegs() << endl;
    else
        cout << pa->getNumLegs() << endl;
}
```

Student Number:

Name:

11. (5 pts) What is the expected output of this program? (If a compile error is expected, just write down "error").

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

class Person {
public:
    void talk()    {cout << "a ";}
};

class Student : public Person {
public:
    virtual void talk()    {cout << "b ";}
};

class CSStudent : public Student {
public:
    void talk()    {cout << "c ";}
};

class Faculty : public Person {
public:
    void talk()    {cout << "d ";}
};
```

```
int main() {
    Person ps;
    Student st;
    CSStudent csst;
    Faculty fc;

    Person* p1 = &ps;
    Person& p4 = fc;
    Student* s2 = &csst;

    p1->talk();
    p4.talk();
    s2->talk();
}
```

12. (7 pts) Add a constructor to the Complex class so that the following outputs are displayed when the program is executed.

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

class Complex {
public:
    Complex() : real(0.0), imag(0.0) {}
    Complex(double r, double i) : real(r), imag(i) {}
    Complex(const Complex& c) : real(c.real), imag(c.imag) {}
    // Add a constructor here

    Complex operator+(const Complex& c) const
    { return Complex(real+c.real, imag+c.imag); }

public:
    double real, imag;
};

int main() {
    Complex a(1.1, 2.2), b(2.2, 5.5), c;
    c = a + b;
    cout << c.real << " " << c.imag << endl;
    c = a + 3.3;
    cout << c.real << " " << c.imag << endl;
    return 0;
}
```

```
3.3 7.7
4.4 2.2
```

Output:

Student Number:

Name:

13. (8 pts) Add an overloaded operator member function to the Point class so that the following outputs are displayed when the program is executed. The added operator must follow the behavior of the corresponding C++ built-in operator of the same kind.

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

class Point
{
private:
    double x, y;

public:
    Point(double x_, double y_):x(x_), y(y_) {}

    // Add an overloaded operator member function here

    void print()
    { cout << "(" << x << ", " << y << ")" << endl; }
};

int main(int argc, char* argv[])
{
    Point p2(1, 2);
    (p2++).print();
    p2.print();

    return 0;
}
```

Output:

```
(1, 2)
(2, 3)
```

14. (6 pts) What is the expected output of this program with the following standard input?

```
#include <iostream>
using namespace std;
double divide(int a, int b) {
    if( b == 0 ) {
        cout << "a ";
        throw "exception";
    }
    cout << "b ";
    return (a/b);
}
int main () {
    int x, y;
    double z;
    cin >> x >> y;
    cout << "c ";

    try {
        z = divide(x, y);
        cout << "d ";
    }
    catch (const char* msg) {
        cout << "e ";
    }
    cout << "f ";
    return 0;
}
```

Standard input:

```
1 2↵
```

Student Number:

Name:

15. (5 pts) Modify the following program to compile successfully and print the output below. You cannot change the name of functions, variables, classes, and class inheritance relationships. You can modify ONLY ONE line of code.

```
1. #include <iostream>
2. using namespace std;
3.
4. template <typename T>
5. class Cls1
6. {
7.     T var1;
8. public:
9.     Cls1(const T& v):var1(v) {}
10.    void foo()
11.    {
12.        cout << var1 << endl;
13.    }
14.};
15.
16.template <typename T>
17.class Cls2 : public Cls1<T>
18.{
19.    T var2;
20.public:
21.    Cls2(const T& v):Cls1(v) {}
22.    void test()
23.    {
24.        Cls1<T>::foo();
25.    }
26.};
27.
28.int main()
29.{
30.    Cls2<int> cls2(10);
31.    cls2.test();
32.
33.    return 0;
34.}
```

Output:

10

Student Number:

Name:

16. (7 pts) What is the expected output of the following program?

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

class CleaningUp{
private:
    string word;
public:
    CleaningUp (const string & str) {
        word = str;
        cout<< "Created: " << word <<endl;
    }
    ~CleaningUp() {
        cout<< "Destroyed: " << word <<endl;
    }
};

void ThrowsException() {
    CleaningUp hi("1");
    int* pi = new int;
    throw "Exception";
    delete pi;
    CleaningUp bye("2");
}

int main() {
    try {
        ThrowsException();
        cout << "3"<< endl;
    }
    catch (const char* e) {
        cout << "4"<< endl;
    }
    return 0;
}
```