

**Instructions:**

- Print your name in the space provided below.
- Answer each question in the space provided. If you need to continue an answer onto the back of a page, clearly indicate that you have done so, and label the continuation with the question number.
- If you want partial credit, justify your answers briefly and concisely, even when justification is not explicitly required.
- Thinking is encouraged.
- There are 20 questions, priced as marked. The maximum score is 100.
- When you have completed the test, sign the pledge at the bottom of this page and turn in the test.
- This is a closed-book, closed-notes examination. No calculators or other electronic devices may be used during this examination. You may not discuss (in any form: written, verbal or electronic) the content of this examination with any student who has not taken it. You must return this test form when you complete the examination. Failure to adhere to any of these restrictions is an Honor Code violation.

Do not start the test until instructed to do so!

Name _____
printed

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

signed

1. [4 points] Which of the following best describes the primary difference between aggregation and association?
 - 1) Association relationships are always maintained by a pointer, but aggregation relationships are never maintained by a pointer.
 - 2) In an association the objects exist independently of one another, but in an aggregation they do not.
 - 3) Aggregation relationships may change but association relationships are unchanging (static).
 - 4) Association relationships always involve objects of the same class, whereas aggregation relationships always involve objects of different classes.
 - 5) There is no difference between association and aggregation.
 - 6) None of these.

 2. [4 points] The use of inheritance directly promotes which of the following software engineering goals?
 - 1) information hiding
 - 2) separation of interface from implementation
 - 3) encapsulation
 - 4) reusability
 - 5) obfuscation
 - 6) None of these.

 3. [4 points] Which of the following are true if the class D is derived from the class B, using public inheritance.
 - 1) Member functions of D can always access all the data members of B.
 - 2) Member functions of D can only directly access the data members of B that are `public`.
 - 3) Member functions of D cannot directly access any of the `private` data members of B.
 - 4) Objects of class D inherit only the `public` and `protected` members of class B.
 - 5) All the members of class B become `public` members of class D.
 - 6) None of these.

 4. [4 points] Which of the following best describes access rights to a function member declared in the protected section of a class B, from which the class D has been derived using `public` inheritance?
 - 1) It may be called from any member function of the class D.
 - 2) It may be called from any function at all, whether it is a member of the class D or even procedural.
 - 3) It may only be called by member functions of the class B.
 - 4) It may be called from member functions of the class D only if those functions in D are `public` or `protected`.
 - 5) It is illegal to declare a member function as `protected`.
 - 6) None of these.
-

For questions 5 through 8, consider the following class, which represents a rational number (ratio of two integers like 1/2):

```
class Rational {
    friend ostream& operator<<(ostream& Out, const Rational& toPrint);
private:
    int Top, Bottom;
public:
    Rational();
    Rational(int T, int B);
    bool operator==(const Rational& RHS) const;
    bool operator<(const Rational& RHS) const;
    bool operator>(const Rational& RHS) const;
    Rational operator+(const Rational& RHS) const;
    Rational operator-(const Rational& RHS) const;
    Rational operator*(const Rational& RHS) const;
    Rational operator/(const Rational& RHS) const;
};
```

5. [5 points] In the class `Rational`, `operator<<` is declared with the keyword `friend`. Explain briefly what is gained by declaring `operator<<` to be a friend of the class.

6. [5 points] Explain briefly the technical reason that `operator<<` could not be a member of the class `Rational`.

7. Consider the following client code:

```
. . .
Rational X(1, 3), Y(1, 4); // Line 1
Rational Z; // Line 2
Z = X + Y; // Line 3
X = X + 1; // Line 4
```

[4 points] When the addition operation on the right hand side of Line 3 is executed, `operator+` is executed within what object?

[4 points] Consider the assignment in Line 3; there is no overloaded assignment operator for the class `Rational`. Does this result in a compile-time error, a run-time error, or a logical error (but not a run-time error), or no error at all?

[4 points] Consider the addition operation in Line 4; does this result in a compile-time error, a run-time error, or a logical error (but not a run-time error), or no error at all?

8. [8 points] Write an implementation for the `Rational` member `operator*`:

```
Rational Rational::operator*(const Rational& RHS) const {

}
}
```

For questions 9 through 11, consider the queue template and main function:

```
// Queue.h
template <class T> class Queue {
private:
    T* L;
    int Sz;
public:
    Queue(int Size = 100);
    T Front() const; // returns 1st elem
    void Push(const T& Data);
    void Pop();      // removes 1st elem
    bool isEmpty() const;
    ~Queue();
};

template <class T>
Queue<T>::Queue(int Size) {
    Sz = Size;
    L = new T[Size];
}
template <class T> Queue<T>::~~Queue() {
    delete [] L;
}
// rest of implementation follows . . .

#include "Queue.h"

bool In(Queue<int> Q, int Target);

int main() {
    Queue    Q1;    // Line 1
    Queue<T>  Q2;    // Line 2
    Queue<int> Q3;    // Line 3
    . . .
    return 0;
}

bool In(Queue<int> Q, int Target) {
    while ( !Q.isEmpty() ) {
        int Next = Q.Front();
        if ( Target == Next )
            return true;
        Q.Pop();
    }
    return false;
}
```

9. [5 points] Which of the declarations in main() are valid?

- | | | |
|----------------|-----------------------|---------------------------|
| 1) Line 1 only | 4) Lines 1 and 2 only | 7) None of them are valid |
| 2) Line 2 only | 5) Lines 1 and 3 only | |
| 3) Line 3 only | 6) Lines 1, 2 and 3 | |

10. [5 points] Assuming, for the sake of this question only, that the declaration of Q3 in main() is valid, clearly describe the unfortunate side effect the following function call would have:

```
bool gotIt = In(Q3, 42);
```

11. [5 points] Assume the queue template were to be distributed to users. Aside from any logical errors in the implementation, the users would have to be supplied with the source for the template implementation, not just with the header file and an associated compiled object (obj) file. Explain why this is so.

For questions 12 through 17, consider the classes and main function:

```
class B {
private:
    int aa;
protected:
    void setaa(int ca) {aa = ca;}
public:
    B(int ia = 0) {aa = ia;}
    int getaa() const {return aa;}
};

class D : public B {
private:
    char cc;
public:
    D(char ic = '+') {cc = ic;}
    char getcc() const {return cc;}
};

// assume needed #includes
int main() {

    D D1('m');           // Line 1
    D D2;                // Line 2
    B B1;                // Line 3
    D1.setaa(5);         // Line 4
    cout << D1.getaa(); // Line 5
    B1 = D1;            // Line 6

    return 0;
}
```

12. [2 points] Given the declaration in Line 1, what is the value of the data member `aa` in the object `D1`?

13. [2 points] Given the declaration in Line 2, what is the value of the data member `cc` in the object `D2`?

14. [3 points] Is the member function call in Line 4 valid? Why or why not?

15. [3 points] Is the member function call in Line 5 valid? Why or why not?

16. [4 points] Is the assignment in Line 6 valid? If yes, describe the contents of the object `B1` after the assignment. If no, explain why not.

17. [5 points] Could the following constructor implementation be added to class D? If not, explain why.

```
D::D(int ia, char ic = 'x') {
    aa = ia;
    cc = ic;
}
```

For questions 18 and 19, consider the following classes and main function:

<pre>class Sat { private: int Fuel; public: Sat(); void chkFuel(); void Fire(); }; Sat::Sat() { Fuel = 1000; } void Sat::chkFuel() { if (Fuel <= 0) throw string("no fuel"); } void Sat::Fire() { if (Fuel > 0) Fuel--; }</pre>	<pre>class Con { private: Sat* pSat; public: Con (); bool satOK(); void Tweak() {pSat->Fire();} }; Con::Con() { pSat = NULL; pSat = new Sat; if (pSat == NULL) throw 42; } bool Con::satOK() { if (pSat == NULL) return false; try { pSat->chkFuel(); } catch (string S) { delete pSat; pSat = NULL; return false; } return true; }</pre>
--	---

```
// assume any necessary #includes
int main() {

    Con myControl;
    while ( myControl.satOK() ) {
        myControl.Tweak();
    }

    return 0;
}
```

18. [4 points] Describe what would happen if the given call to the function `Control::satOK()` occurred and the value of the data member `Fuel` of `myControl`'s `Sat` object were 0.

19. [4 points] Describe what would happen if the dynamic allocation operation in the `Control` constructor failed.

20. [12 points] Consider the system description below:

A kennel must provide space and care for a variety of animals. Most dogs are housed in "runs" which are connected to a fenced space outside the kennel building. Small dogs, and cats, are housed in small to medium cages. Other types of animals are cooked and eaten. (Sorry, couldn't resist.) Large dogs, such as golden retrievers and Newfoundland retrievers, must be kept in very large runs and exercised twice a day. Medium dogs, such as beagles and spaniels, must be housed in average-sized runs and exercised once a day. Small dogs, such as Shih Tzu and Chihuahuas, are housed in medium cages and exercised once a day.

From this description, identify a set of potential classes, representing residents of the kennel not cages or runs, that should be related by inheritance, and draw the inheritance hierarchy for those classes below. Each base class should be shown above any classes derived from it. It is not necessary, or desirable, to show class details. You should include all relevant classes that are directly implied by the description above.