Instructions:

- Print your name in the space provided below.
- Answer each question in the space provided.
- If you want partial credit, justify your answers briefly and concisely, even when justification is not explicitly required.
- There are 11 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
class Rational {
    friend istream& operator>>(istream& In, Rational& toPrint);
    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+(int RHS) const;
    Rational operator-(const Rational& RHS) const;
    Rational operator-(int RHS) const;
    Rational operator*(const Rational& RHS) const;
    Rational operator*(int RHS) const;
    Rational operator/(const Rational& RHS) const;
    Rational operator/(int RHS) const;
};

1. [6 points] Assuming all of the member functions of the class Rational have been implemented correctly, circle any of the following statements that would not compile.

    Rational R1(1,1), R2(2,1), R3(3,1), R12(1,2), R13(1,3), R23(2,3);

    Rational SumR, SubR, ProdR, QuotR;

    SumR = R1 + R2 + R3 + R12;

    SubR = 3 * R2 - R23;

    ProdR = R3 * R13 + 4;

    QuotR = R12 / Rational(2, 5);

2. [10 points] The mathematical property of commutativity between integers and rationals is not supported on the standard arithmetic operation of multiplication by the C++ Rational class. Give the implementation of the multiplication operator that would need to be added to support commutativity.
For the next three questions, consider the following template classes, which represents a Queue with an underlying double linked list:

```cpp
// DNodeT.h
#ifndef DNODET_H
#define DNODET_H

template <class Item> class DNodeT {
public:
    Item* Data;
    DNodeT<Item> *Next, *Prev;

    DNodeT();
    DNodeT(Item const *newData);
};

// . . . DNodeT template implementation goes here
#endif

// QueueT.h
#ifndef QUEUET_H
#define QUEUET_H
#include "DNodeT.h"

template <class Foo> class QueueT {
protected:
    DNodeT<Foo> *Front, *Rear; //Head of list == Front of Queue
public:
    QueueT();
    QueueT(const QueueT<Foo>& Source);
    void Enqueue(Foo* const Item);
    Foo* Dequeue();
    bool isEmpty() const;
    QueueT<Foo>& operator=(const QueueT<Foo>& Source);
    ~QueueT();
};

// . . . QueueT template implementation goes here
#endif
```

3. [8 points] Consider a program that declares the following object: `QueueT<string> todoList;`

Is the relationship between `QueueT` and `string` (not `string*`) an association, aggregation or inheritance? Justify your answer.
4. [6 points] What provision, if any, is made in the queue or node implementations given above to support the derivation of new classes from those?

5. [6 points] What provision, if any, is made in the queue or node implementations given above to support storing a polymorphic data type?

6. [8 points] The designer of a graphical window class for a graphical user interface library is faced with a decision. Certain attributes (e.g., border style) should be the same for all instances of the class, but should also be dynamically changeable. What C++ feature should she use to handle this?
7. [12 points] Consider the two classes below. The designer is faced with one annoying problem. Given the logical significance of the class B, client code should not be able to modify the data member X of B.

```cpp
class A {
    protected:
    string X;

    public:
    A(const string& iX) {X = iX;}
    string getX() const {return X;}
    void setX(const string& iX) {X = iX;}
};

class B : public A {
    public:
    B(const string& iX) : A(iX) {}
};
```

Aside from abandoning the use of inheritance, describe one way the designer could deal with this problem, and describe any shortcomings of your solution.

8. Consider the template mechanism in C++.
   (a) [6 points] Explain why the source code for a template will not compile.

   (b) [6 points] Explain briefly, in general terms, how the fact stated in (a) is dealt with (since we can, in fact, write useful template implementations and incorporate them in our programs).
9. [12 points] A contemporary C++ programming textbook presents the design of a class called Insurance, with the following attributes and behaviors:

- An insurance policy number
- The name of the person who owns the policy
- The annual premium
- Accessors for all the data members
- Mutators for all the data members
- Constructors/destructors as needed

The textbook then suggests (as an exercise) "declare two derived classes called Automobile and Home that inherit the Insurance base class". Assume that the proposed name of the class Automobile is an accurate name for the intended abstraction. In no more than 100 words, critique the suggestion that Automobile be derived from Insurance. If you argue that this is not a good idea, explain why not and suggest a reasonable alternative. If you argue that this is a good idea, explain why.
10. Farey numbers are a mathematical construct that somewhat resemble rational numbers. The primary differences are that addition of Farey numbers follows the rule
\[
\frac{a}{b} + \frac{c}{d} = \frac{a+c}{b+d}
\]
and that the other standard arithmetic operations are not (usually) defined.

(a) [8 points] Assuming that the Rational class presented earlier has been implemented, consider deriving a class \textit{Farey} from \text{Rational}. Are there any particular reasons that would be difficult or clumsy?

(b) [6 points] Would those reasons still apply if the hierarchy were reversed. That is, what if a complete \textit{Farey} class were implemented first, and then a \text{Rational} class were derived from it? Explain.

11. [6 points] What are the three elements that must all be present in order to achieve polymorphism in C++?