

**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 14 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

For the next two questions, 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+(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 = 4 + R1;
```

```
SubR = R2 - R2;
```

```
ProdR = R3 * R13;
```

```
QuotR = R12 / 5;
```

2. [10 points] Is the mathematical property of commutativity between integers and rationals supported on the standard arithmetic operation of addition by the above C++ `Rational` class? If not, give the prototype and implementation of the addition 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:

```
// LinkNodeT.h
#ifndef LINKNODET_H
#define LINKNODET_H

template <class Item> class LinkNodeT {
private:
    Item Data;
    LinkNodeT<Item> *Next, *Prev;

public:
    LinkNodeT();
    LinkNodeT(Item newData);
    void setData(Item newData);
    void setNext(LinkNodeT<Item>* newNext);
    void setPrev(LinkNodeT<Item>* newNext);
    Item getData() const;
    LinkNodeT<Item>* getNext() const;
    LinkNodeT<Item>* getPrev() const;
};

// . . . LinkNodeT template implementation goes here

#endif

// QueueT.h
#ifndef QUEUE_T_H
#define QUEUE_T_H
#include "LinkNodeT.h"
template <class Foo> class QueueT {
private:
    LinkNodeT<Foo> *Front, *Rear; //Head of list == Front of Queue
public:
    QueueT();
    QueueT(const QueueT<Foo>& Source);
    void Enqueue(const Foo& Item);
    Foo Dequeue();
    bool isEmpty() const;
    QueueT<Foo>& operator=(const QueueT<Foo>& Source);
    ~QueueT();
};

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

3. [5 points] Is the relationship between QueueT and LinkNodeT an association, aggregation or inheritance? Justify your answer.

4. [10 points] Assume that all of the functions of both classes have been implemented, except for the copy constructor and assignment operator overload. Give the C++ code to implement the assignment operator overload according to the above class declarations:

```
QueueT<Foo>& operator=(const QueueT<Foo>& Source) {
```

5. [6 points] For the above QueueT template class, briefly explain the implementation differences between the assignment operator overload and the copy constructor?

For the next two questions, consider the following classes:

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

class Point {
private:
    double X, Y;
public:
    Point(double iX = 0.0, double iY = 0.0) {X = iX; Y = iY;}
    double getX() const {return X;}
    double getY() const {return Y;}
    void setPoint(double iX = 0, double iY = 0) {X = iX; Y = iY;}
    ~Point() {}
};

const double PI = 3.14159;

class Circle {
private:
    Point Center;
    double Radius;
public:
    Circle(Point C = Point(4.2, 3.1), double iR = 0.5) {Center = C; Radius = iR;}
    Circle& setCenter(Point C) {Center = C; return *this;}
    Circle& setRadius(double iR) {Radius = iR; return *this;}
    Point getCenter() const {return Center;}
    double getRadius() const {return Radius;}
    double Area() const {return (PI * Radius * Radius);};
    ~Circle() {}
};
```

6. [4 points] If one makes the following declaration:

```
Circle X;
```

What is the center of X or is it random?

7. [9 points] Given the following code segment to execute:

```
Point P(7, 11);
Circle Hoop(P, 3.14);
Point Q(0, 3);
Hoop.setCenter(Q).setRadius(2.72);
cout << Hoop.getCenter().getX() << " " << Hoop.getCenter().getY() << endl;
cout << Hoop.getRadius() << endl;
```

What is output by the above code?

For the next 5 questions, consider the following class, which builds upon the classes from the previous two questions:

```
#include <cmath>

class Cone : public Circle {
private:
    double Height;
public:
    Cone(Circle Base, double iH) : Circle(Base) {Height = iH;}
    Cone& setHeight(double iH) {Height = iH;}
    double getHeight() const {return Height;}
    double SlantHeight() const {
        return (sqrt(getRadius() * getRadius() + Height * Height));
    }
    double Area() const {
        return (PI * getRadius() * getRadius() +
            PI * getRadius() * SlantHeight());
    }
    double Volume() const {return (PI * getRadius() * getRadius() * Height / 3.0);}
    ~Cone() {}
};
```

8. [8 points] Can one make the following declaration?

```
Cone A[10];
```

Briefly explain?

9. [8 points] Given the following declaration:

```
Cone Z(Circle(Point(3,3), 3.14159), 2.71828);
```

Can one make the following member invocation, (briefly explain)?

```
Point pc = Z.getCenter();
```

For the next 2 questions, consider the following alternative implementation of `Cone::Volume()`:

```
double Volume() const {
    return (PI * Radius * Radius * Height / 3.0);
}
```

10. [8 points] Explain why this alternative implementation is not valid, (the above formula is OK).

11. [8 points] Aside from eliminating inheritance, what change would make this alternative implementation valid, (without changing the alternative implementation code itself)?

12. [8 points] Briefly explain how Base is being used in the constructor for Cone?

For the next two questions, consider the following function:

```
void check() throw(exception) {
    try {
        throw exception(); //throw a standard exception object
    }
    catch (exception& e) {
        cout << e.what() << endl;
        check();
        throw(e);
    }
}
```

13. [6 points] Assume that check has been called from main(). Given that the standard exception class provides a member function what() that returns a string describing the exception error, which of the following statements about the above function is true?

- 1) The code will not compile since it is illegal to throw an exception within a catch clause.
- 2) The code will not compile since it is illegal to throw an exception before the function body.
- 3) If the function that invokes check(), (i.e. main()) does not catch the exception the program will abort.
- 4) The invocation of check() will be removed from the runtime stack as soon as the exception is thrown.
- 5) The invocation of check() will result in infinite recursive calls to check() being made.
- 6) None of the above, they are all false statements.

14. [4 points] How many times will the throw(e); statement in the above function be executed?

- | | |
|------|-----------------------|
| 1) 1 | 4) infinite, ∞ |
| 2) 2 | 5) 0 |
| 3) 3 | 6) \leq INT_MAX |