CS 7A Final Exam Solutions - Spring 2015 - Final Exam 5/19/15

Write responses on separate paper.

1. Define each of the following terms in the context of this course. Give an example of how each may be used.

   (a) *conditional statement*
   ANS: A statement that resolves to a `true` or a `false` result. Also, The ?: construct is called an arithmetic if or a conditional expression. The value of (a¿=b)?a:b is a if a¿=b and b otherwise.

   (b) *for statement*
   ANS: Iterating over a sequence of numbers is so common that C++, like most other programming languages, has a special syntax for it. A **for-statement** is like a while-statement except that the management of the control variable is concentrated at the top where it is easy to see and understand.

   (c) *while statement*
   ANS: The language construct we used is called a while-statement. Just following its distinguishing keyword, while, it has a condition "on top" followed by its body:

```
1 while (i<100) // the loop condition testing the loop variable i
{
3         cout << i << '\t' << square(i) << '\n';
          ++i ; // increment the loop variable i
5 }
```

   The loop body is a block (delimited by curly braces) that writes out a row of the table and increments the loop variable, `i`. We start each pass through the loop by testing if `i<100`. If so, we are not yet finished and we can execute the loop body. If we have reached the end, that is, if `i` is 100, we leave the while-statement and execute what comes next. In this program the end of the program is next, so we leave the program.

   (d) *operator precedence*
   ANS: Examples of operators are `+,-,*,/,%`, etc. Operator precedence refers to the order in which the operators are implemented when used together.

   (e) *type*
   ANS: Something that defines a set of possible values and a set of operations for an object.

2. Consider the program shown below:

```
1 int main() {
      ofstream ofs("data.txt");
3     cout << "\nEnter␣four␣positive␣integers␣followed␣by␣ctrl+z:␣";
      int x{0};
5     while(cin>>x) ofs<<x<<"␣";
      ifstream ifs("data.txt");
7     int a{0};
      ofs.close();
9     while(!ifs.eof()) {
          ifs>>x>>a;
11        while (x >= 2) {
              cout << a << "␣";
```

```
13              x = x - 1;
                if (x % 2 == 0) {
15                  a = a + x;
                }
17              else {
                    a = a - x;
19              }
            }
21          cout << endl;
        }
23  }
```

Suppose the user responds to the prompt like so:

```
Enter four positive integers followed by ctrl+z:  5 0 7 0 ^Z
```

(a) What will become of the file `data.txt` ?
`data.txt` will contain these four numbers like so

```
5 0 7 0
```

(b) Show the list of values taken by the variable `a` that are output to the console as the program executes.

`x`	`a` (as output to console)
5	0
4	4
3	1
2	3
1	2
⋮	⋮
7	0
6	6
5	1
4	5
3	2
2	4

3. It is known that $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$. The program below uses this to approximate $\sin(x)$ for user-supplied values of x:

```
1  double sine(double x) {
       const double twoPi = 6.28318530718;
3      double sum{0};
       double tolerance{ 1e-3};
5      double n{0};
       while(abs(x)>twoPi) x -= twoPi;
7      double next = x;
       do {
9          sum += next;
           n += 2;
11         next *= -x*x/(n*(n+1));
           cout << "\nnext = " << next << ", sum = " << sum;
13     } while(abs(next)>tolerance);
       return sum;
15 }

17 int main() {
       double x;
19     cout << "\nEnter a value for x to compute sin(x): ";
       while(cin>>x) {
21         cout << "\nsin(" << x << ") = " << sine(x);
       }
23 }
```

(a) Provide a definition for `abs()` to take a double and return its absolute value.
ANS:

```
1  double abs(double x) {
           return x>0?x:-x;
3  }
```

(b) What does the while loop on line 6 do? Why?
ANS: If $x > 2\pi$ keep subtracting 2π until you get a value between 0 and 2π with the same value of sin(). This is done to increase the rate of convergence.

(c) The while loop on on line 6 will not behave properly for some negative x. What's wrong? How would you fix it?
For values of $x \leq -2\pi$, the code keeps subtracting 2π, leading to an infinite loop. To fix this, change the body of the while loop to test whether `x` is positive or negative. If it's positive and greater than 2π, subtract 3π if it negative with $|x| > 2\pi$, add `twoPi`.

(d) The series formula for $\cos(x)$ is $\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$. Modify the `sine()` function to produce a `cosine()` function.
ANS:

```
1  double cosine(double x) {
       const double twoPi = 6.28318530718;
3      double sum{1};
       double tolerance{ 1e-5};
5      double n{1};
       while(abs(x)>twoPi) {
7          if(x>0) x -= twoPi;
```

```
        else x += twoPi;
    }
    double next = -x*x/2;
    do {
        sum += next;
        n += 2;
        next *= -x*x/(n*(n+1));
        cout << "\nnext = " << next << ", sum = " << sum;
    } while(abs(next)>tolerance);
    return sum;
}
```

4. Help complete the design and implement a rational number class, `Rational` started below by answering the questions that follow. Addition, and multiplication operator overloading are provided.

```
int gcd(int x, int y) {
        // greatest common divisor using Euclid's algorithm
        x = abs(x);     // don't get confused by negative values
        y = abs(y);
        while (y) {
                int t = y;
                y = x%y;
                x = t;
        }
        return x;
}
//--- the class ----------------------------------------------------------------
class Rational {
public:
        Rational(int n, int d) :num(n), den(d) { normalize(); }
        Rational() :num(0), den(1) { }

        void normalize() {// keep denominator positive and minimal
                if (den==0) error("negative denominator");
                if (den<0) { den = -den; num = -num; }
                int n = gcd(num,den);
                if (n>1) { num/=n; den/=n; }
        }
        int num, den;
};
Rational operator+(Rational x1, Rational x2)
{
        Rational r(x1.num*x2.den+x1.den*x2.num, x1.den*x2.den);
        r.normalize();
        return r;
}
Rational operator*(Rational x1, Rational x2)
{
        Rational r(x1.num*x2.num,x1.den*x2.den);
        r.normalize();
        return r;
}
ostream& operator<<(ostream& os, Rational x) {
```

```
        return cout << '(' << x.num << '/' << x.den << ')';
40 }
```

(a) Explain what the `gcd()` function does and how it is used here.
ANS: `gcd()` is a function that implements the Euclidean algorithm to compute and return the greatest common divisor of the two inputs.

(b) Describe the constructor function for the `Rational` class. What parameters are passed? What is the initializer list? What happens in the body of the function? What would the statement `Rational r2(40,24);` do?
ANS: This call to the constructor would set `num` to 40 and `den` to 24 and then call `normalize` which would compute the `gcd(40,24)=8` and divide both `num` and `den` by 8 to produce the reduced form of the rational number `5/3`.

(c) Write functions to overload the subtraction(-) and division(/) operators.
ANS:

```
Rational operator-(Rational x1, Rational x2) {
2       Rational r(x1.num*x2.den-x1.den*x2.num, x1.den*x2.den);
        return r;
4 }
Rational operator/(Rational x1, Rational x2) {
6       Rational r(x1.num*x2.den,x1.den*x2.num);
        return r;
8 }
```