Comp 170 Exam 3 Overview.

Exam Ground Rules

The exam will be closed book, no calculators. You may bring notes on three sides of 8.5x11 inch paper (for example three pieces of paper, if you write on only one side). You may start early or work late, like the last two exams.

This is your last exam in a cumulative course. It counts more than the last two exams. Not a lot is new: fancier array index manipulations, Lists, Dictionaries, interfaces, and creating and using instance variables. Make sure you have solidified the ideas from previous exams.

Main topics that may be on exam 3: All the notes, assignments, and class discussions, with some emphasis on Chapters 11-13, 15 (skipping Ch 14).

1. Follow code to play computer, including arbitrary use of collections, loops or nested loops, decisions, instance methods, static functions.
2. Work with common types and the standard conversions between them.
3. Recognize and write code with format string involving both field widths and precision; string.Format.
4. Be familiar with the library methods from the first two exam reviews.
5. Be able to read, follow, and write code using Lists, including constructors, reading and assigning, property Count, and methods Add, Remove, RemoveAt, Contains.
6. Be able to read, follow, and write code using Dictionaries, including constructors, reading and assigning, and the property Keys.
7. Lists and Dictionaries give us more examples of collections, beyond arrays. Expect to write lots of loops iterating through and using all these kinds of collections.
8. Read and write object code, with constructors and instance methods, including methods where objects of the receiver's type are parameters. Use **this** refer to the current object when needed.
9. Work with text files: opening, closing, reading, writing, detecting end of file.
10. Use interface types, write an interface.
11. Use and write ToString methods that work implicitly for printing and concatenation.

Get clear on the problem statements and ask for help if necessary on solutions ***before*** looking at the solutions at the end!

The same suggestions as for the last exam apply on review and doing review problems. This exam will involve more coding and less “What does this print?” than the last exam.

Sample Review Problems for Exam 3

1. What is printed by this code fragment?

```
int a = (int)((5/3)*4.0)
int b = (int)((8.88/4)*10)
Console.WriteLine(a + " " + b);
```

2. Is this a legal interface? Explain.

```
interface X3I {
    int S2L(string s);
    string I2S(int x)
    {
        return ""+x;
    }
}
```

3. What is needed to make this class satisfy the (corrected) interface X3I of problem 2?

```
public class X3 {
  public X3() {}
  public int S2L(string s)
  {
      return s.Length;
  }
  public string I2S (int x)
  {
      return ""+x;
  }
  public void hi()
  {
    Console.WriteLine("Hi");
  }
}
```

4. Assuming Problem 3 is modified to be legal, explain which of these statements is legal:
a. `X3I n = new X3();`
b. `X3I x = new X3I();`

5. Consider the following code fragment:

```
var m=new Dictionary<string,string>();
m["a"] = "stuff";
m["b"] = "junk";
m["c"] = "garbage";
Console.WriteLine(m["b"]);
```

a. What is printed?
b. What are the elements of the `m.Keys`?

6. Write the C# code for an interface Bar which includes just one method, with signature:

```
public void Foo(int x)
```

7. What is printed?

```
for (int k = 1; k < 4; k++)  {          //1
  for (int m = k; m < 4; m++) {         //2
    Console.Write((10*k +m) + " ");     //3
  }
  Console.WriteLine();                  //4
}
```

8. What does the Test program display? (Foo below)

```
class Test
{
  public static void Main()
  {
    Foo f1 = new Foo(3, 5);                    //1
    Foo f2 = new Foo(2, 10);                   //2
    Console.WriteLine("f1: {0} f2: {1}",
                        f1, f2);               //3
    f2.Pah(f1);                                //4
    Console.WriteLine("f1: {0} f2: {1}",
                       f1, f2);                //5
  }
}

class Foo
{
  private int x, y;

  public Foo(int a, int b)         //6
  {
    x = a;                         //7
    y = b;                         //8
  }

  public string ToString()
  {
    return  "x:"+x+ " y:" +y;      //9
  }

  public void Pah(Foo f)           //10
  {
    x = x + f.y;                   //11
    y = x*f.x;                     //12
  }
}
```

9. Remember the Fraction class with int instance variables num and denom. Complete the Abs method for the class Fraction. Math.Abs is the corresponding function defined for integers.

```
/**Return a new Fraction whose value is
the absolute value of this Fraction.*/

public Fraction Abs(int n)
```

10. Recall the first two methods below from the review for Exam 2. *Complete* the third method, NMatch.

```
public static void ASqr(int x[])
// squares the elements of x:
// For example if x initially contains 2, 3, 5, 7,
//        then at the end x contains  4, 9, 25, 49.

public static int AProd(int x[])
// returns the product of the elements of x (or 1 if there are no elements):
// For example if x initially contains 2, 3, 5,
//    then aProd returns 30.

public static int NMatch(int x[], int y[])
// returns the number of matches of elements of x and y
//    with the same subscript:
// For example if  x contains 2, 3, 5, 7, 9
//                 y contains 2, 7, 5, 8, 9, 6, 3
//    then nMatch returns 3  (2's, 5's, 9's, not 7's or 3's)
```

11. Consider the class P11 with an instance variable x which is an int array. First write a constructor that takes an array parameter and makes x be a *copy* of the parameter (not an alias). Then modify the static methods of problem 10 to be instance methods with the headings below.

```
class P11
{
      private int[] x;

      public P11(int[] a) // creates x and copies the elements of a into it
      public void ASqr() // replace elements of x by their squares
      public int AProd() // return product of elements of x
      public int NMatch(P10 y)//count matches between this.x and y.x
      ...
}
```

12. Complete the code for the method ReadFractions, below. Suppose the reader parameter is connected to a text file containing pairs of integers, which are intended as numerators and denominators of fractions. For instance, if reader is connected to a file containing

```
2 5
-2 11
8 3
```

then the method should return a new List of Fractions representing 2/5, -2/11, and 8/3.

```
public static List<Fraction> ReadFractions(StreamReader reader)
```

13. Complete the method Shifted, which takes an array v as parameter and returns a new array of the same size, where the last element of v becomes the first element of the returned array, and all the other elements of v are shifted one index in the returned array. Examples showing the elements of v and the elements of the returned array:

```
          v:  1, 2, 3, 4, 5              v: 2, 4, 6, 8              v:  2, 5            v:  7
   returned:  5, 1, 2, 3, 4      returned:  8, 2, 4, 6       returned:  5, 2     returned:  7
```

```
public static int[] Shifted(int[] v)
```

14. Write a function to print a triangle of asterisks ('*'), starting with one asterisk, building up to n asterisks.

```
Examples:    for n=4:   *              n=2:  *              n= 1: *
                        **                   **
                        ***
                        ****
```

```
public static void PrintTriangle(int n)
```

// SOLUTIONS ON NEXT PAGE

Exam 3 Review problem solutions

1. 4 22

a = (int)(1*4.0) = (int)(4.0) = 4 (initial *integer* division)

b = (int)(2.22*10) = (int)(22.2) = 22

2. no function bodies allowed. Replace

```
  string I2S(int x) {
        return ""+x;
    }
```

by: `string I2S(int x);`

3. heading line needs at the end:

```
  : X3I
```

The extra method is OK.

4a. legal: interface type is an alternate type for any X3.

b. Interfaces have no constrctor. You can only create a object of a class that *implements* the interface.

5. a. `junk` b. `"a", "b", "c"` (order not specified)

6.

```
interface Bar
{
  void Foo(int x);
} // note the semicolon above!
```

7.

```
 11 12 13
 22 23
 33
```

```
Line  k  m  comments
1     1  -  initialize k; k<4 true
2        1  initialize m 1 < 4; true
3           print 1*10+1 = 11
2        2  1+1=2; 2<4 true
3           print 1*10+2=12
2        3  1+2=3; 3<4 true
3           print 1*10+3 = 13
2        4  1+3=4; 4<4 is false, done with inner loop
4        -  print newline
1     2     1+1=2; 2<4 true
2        2  initialize m; 2<4 true
3           print 2*10+2=22
2        3  1+2=3; 3<4 true
3           print 2*10+3 =23
2        4  1+3=4; 4<4 is false, done with inner loop
4        -  print newline
1     3     2+1=3; 3<4 true
2        3  initialize m; 3<4 true
3           print 3*10+3=33
2        4  1+3=4; 4<4 is false, done with inner loop
4        -  print newline
1     4     3+1=4; 4<4 false, done with outer loop
```

8.

```
f1: x:3 y:5 f2: x:2 y:10
f1: x:3 y:5 f2: x:7 y:21
```

line by line:

method	Test:Main----				Foo:Foo		Foo:Bar		
object	f1		f2				f		
variable	x	y	x	y	a	b	x	y	
line									comments
1									invoke Foo
6					3	5			
7	**3**				3	5			new Foo will be called f1, so I write to that object, assigning to x
8	**3**	**5**			3	5			similarly, for y
2									invoke Foo
6	3	5			2	10			
7	3	5	**2**		2	10			new Foo will be called f2, so I write to that object, assigning to x
8	3	5	**2**	**10**					similarly, for y
3	3	5	2	10					calls ToString for f1, f2 returning "x:3 y:5" and "x:2 y:10"; inserts f1:, and f2: before printing
4.	3	5	2	10					to method Pah with receiver f2
10	3	5	2	10			3	5	f is an alias for f1. Since the receiver is f2, x and y refer to f2.x and f2.y
11	3	5	**7**	10			3	5	2 + 5 = 7 to x
12	3	5	7	**21**			3	5	7*3 = 21 to y (using new x!)
5	3	5	7	21					calls ToString for f1, f2 returning "x:3 y:5" and "x:7 y:21"; inserts f1: and f2: before printing

9.

```
{
  return new Rational(Math.Abs(num), denom)
}
```

// or other variations with if statements

10.

```
static int NMatch(int[] x, int[] y)
{
   int matches = 0;
   int n = x.Length;            //OR:  int n = Math.Min(x.Length, y.Length);
   if (n > y.Length) {          //       and then the if statement is not needed
     n = y.Length;
   }
   for (int i = 0; i < n; i++)
     if (x[i] == y[i])
       matches++;
   return matches;
}
```

11.

```
public P11(int[] a)
{
  x = new int[a.Length];
  for (int i = 0; i < x.Length; i++)
    x[i] = a[i];
}

public void ASqr()
{//no changes in body from exam 2 review
   for (int i = 0; i <  x.Length; i++)
     x[i] = x[i]*x[i];
}
```

```
public int AProd()
{ //no body change from exam 2 review
   int prod = 1;
   for (int i = 0; i < x.Length; i++)
     prod = prod*x[i];
   return prod;
}

public int NMatch(P11 y)
  //just replace y by y.x everywhere in problem 10, so
  //y[i] is replaced by y.x[i];
  //y.Length is replaced by y.x.Length
```

12.

```
{
  List<Fraction> list = new List<Fraction>();
  while (!reader.EndOfStream) {
    string[] f = reader.ReadLine().Split(' ');
    list.Add(new Fraction(int.Parse(f[0]), int.Parse(f[1]));
  return list;
}
```

13.

```
{
  int[] a = new int[v.Length]
  a[0] = v[v.Length - 1]
  for (int i = 1; i < v.Length; i++)
      a[i] = v[i-1];
  return a;
}
```

```
//OR using modular arithmetic for the shift
{
..int n = v.Length;
  int[] a = new int[n]
  for (int i = 0; i < n; i++)
      a[(i+1) % n] = v[i];
  return a;
}
```

14.

```
{
  for (int i = 1; i <= n; i++) {
    for (int j = 0; j < i; j++)
      Console.Write('*');
    Console.WriteLine();
  }
}
```