# Comp 170 Exam 2 Overview.

**Exam Ground Rules**

The exam will be closed book, no calculators. You may bring notes on two sides of 8.5x11 inch paper (either both sides of one sheet, or two sheets written on single sides). Again you may stay an hour late.

**Main topics that may be on exam 2: Notes through One Dimensional Arrays, 10.1, plus using arrays as in the last homework and lab 10.6, plus corresponding class discussions.**

1. Material from exam 1. In particular be able to use/find/remember the string class methods again.
2. Follow (play computer on) code with arrays.
3. Write code that involves simple sequential reading or writing to arrays.
4. Follow (play computer on) combinations of if statements, while and for loops.
5. Simple and compound if statements (reading deeply nested ones, writing simpler ones)
6. Read all kinds of loops: while, foreach, for.
7. Write whatever kind of loops work for you – that could be all while loops.
8. Write new combinations of if statements and loops (your choice of loop structure), including nested loops. Interactive loops.
9. Operators ++ (increment ), -- (decrement), +=, -=, …;
10. Basic types: string, int, char, bool, and double; casts between numerical types.
11. Read and write with files and keyboard/screen. You may assume you have access to the UI class that we wrote, with methods PromptLine, PromptInt, and PromptDouble.
12. Files: Opening, closing, check for the end of files.
13. Method Next(int low, int past) for class Random.
14. You can assume we are using all needed namespaces: you do not have to remember or add them.
15. Format notation inside braces for field width and for number of places shown beyond the decimal point in doubles.

**How the topics get used:**

1. Playing computer will still be important, with more complicated code than you would be asked to write.
2. You will likely be asked to write more significant bits of code than on Exam 1. Loops and arrays will likely be prominent, mixed in with if statements....

The recommended text exercises provide further practice.

**Read the following before looking at either the problems or the solutions (same as first exam)!**

1. Study first, compiling notes, and then look at the sample problems. Look at the list at the top of the page and start by filling in any holes. The sample problems cannot give complete coverage, and if you look at them first, you are likely to study just these points first, and will not get an idea how well you are prepared in general.
2. Do not look at the answers until you have fully studied and tried the problems and gotten *help* getting over rough spots in the problems if you need it! Looking at the answers before this time makes the problems be just a few more displayed examples, rather than an opportunity to actively learn by doing and check out where you are. The *doing* is likely to help you be able to *do* again on a test.

1. What is printed by this code fragment?

```
int x = 1;                 //1
while(x < 4) {             //2
  Console.WriteLine(x*x); //3
  x++;                     //4
}
```

2. What is printed by this code fragment?

```
int x = 2;                  //1
while (x < 8) {             //2
  if (x % 2 == 0) {         //3
    x--;                    //4
  } else {
    x += 3;                 //5
  }
  Console.Write(" " + x);  //6
}
```

3. What is printed by the following fragment?

```
int[] v = {5, 3, -2, 1};                   //1
int n = 0;                                 //2
for (int k = 1; k < v.Length; k++) { //3
   n += k*v[k];                            //4
   Console.Write(" " + n);                 //5
}
```

4. What is printed by this program?

```
class SX2A {
  public static void Main() {
    string[] list = {"up", "down", "on", "off"};
    Console.WriteLine(Foo(list));
   }

  public static string Foo(string[] a)
  {
    string s = ""
    for(int i = a.Length-1; i>= 0; i--){
       s += a[i] + a[i];
    }
    return s;
  }
}
```

5. Complete the function definitions:

```
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 (2*3*5).
```

6. Complete the code for a static method CountErrors, that counts all the strings in array e that contain the substring "error", and returns the count.

```
public static int CountErrors(string[] e)
```

7. Complete the static method below, which shows the results of repeatedly dividing an integer n by k, with integer division, stopping at 0. Place a blank before each number printed. Examples:
Output for `KeepDividing(6, 2)` is `6 3 1 0` Output for `KeepDividing(66, 3)` is `66 22 7 2 0`

```
public static void KeepDividing(int n, int k)
```

8. Complete the static method `RandomCount` that uses rand to generate a sequence of n random numbers from 0 through x-1 inclusive, and counts how many of the numbers are less than k.
If `RandomCount(8, 6, 4, rand)` were to happen to generate the random sequence 7, **2**, **1**, 5, **0**, **1** from `rand`, then the method would return `4`.

```
public static int RandomCount(int x, int n, int k, Random rand)
```

9. Complete the definition of the method.

```
public static void EvenList(int n)
// prints out all the even numbers between 0 and n, including 0 and n, starting with 0, each with one space before it.
// Note n may be negative. For example EvenList(6) displays 0 2 4 6    EvenList(0) displays 0
//  EvenList(3) displays 0 2   EvenList(-4) and EvenList(-5) display  0 -2 -4
```

10. Write a Main program that finds the winner in a vote. The program starts by printing "Enter votes". Suppose people take turns entering "y" or "n", and when everyone has voted, someone enters "q". The program ends by saying either "Yes won", "No won" or "A tie". You may assume the only answers given are "y", "n" or "q". (No error testing is needed.). Three sample runs:

```
Enter votes:        Enter votes:        Enter votes:
y                   n                   y
n                   n                   n
y                   y                   n
y                   q                   y
y                   No won              q
q                                       A tie
Yes won
```

```
public static void Main()
```

11. Suppose `reader` is reading from a file . Complete the code fragment so every other line is printed, starting with the first line, and so the file gets closed.

```
StreamReader reader = new StreamReader("someData.txt");
```

12. Complete the definition of the method RepCounts, which returns a new array with the same length as chars, containing the number of repetitions in source of each individual character in chars. Case matters. For example, if source is "Hello there!" and chars is "oeah", then return an array containing {1, 3, 0, 1}.

public static int[] RepCounts(string source, string chars)

13. What is printed?

```
int[] a = { 9, 5, 8, 6};
int i = 2;
Console.WriteLine(2*a[i] + " " + a[i+1] + " " + (a[i-1] - a[i-2]));
```

14. What is printed by the following fragment?

```
double x = 2.2, y=4.44, z=8.888;
Console.WriteLine("12345678901234567890");
Console.WriteLine("{0,6:F2}{1,6:F2}{2,6:F2}", x, y, z);
```

15. What is printed by this code fragment?

```
int a = (int)((5/3)*4.0)
int b = (int)((8.88/2)*20)
Console.WriteLine(a + " " + b);
```

16. What is printed by this code fragment?

```
for (int a = 1; a < 4; a++) {         //1
   for (int b = 5, b < 7; b++) {      //2
        Console.Write(a + " " + b);   //3
   }
}
```

17. Complete this code (a bit of a challenge to do concisely; see the hint):

```
/// print nested boxes with the characters in s; a single copy of the last character is innermost.
/// nest("abc") prints:   nest("12") prints:    nest("x") prints:
///             aaaaa                111                  x
///             abbba                121
///             abcba                111
///             abbba                          // Hint: How many rows and columns?  The s index is given by the
///             aaaaa                          // minimium distance to any edge; you can use Math.Min repeatedly.
static void nest(string s)                                          // SOLUTIONS ON NEXT PAGE
```

## Exam 2 Review problem solutions

1:
```
1
4
9
```

| line | x | comment |
|---|---|---|
| 1 | 1 | |
| 2 | | 2<4 true |
| 3 | | print 1*1 = 1 |
| 4 | 2 | 1+1=2 |
| 2 | | 2<4 true |
| 3 | | print 2*2 = 4 |
| 4 | 3 | 2+1=3 |
| 2 | | 3<4 true |
| 3 | | print 3*3 = 9 |
| 4 | 4 | 3+1=4 |
| 2 | | 4<4 false |

2: `1 4 3 6 5 8`

Details of each time through the loop: init x = 2
2 < 8 so do body of loop: even x, so decrement x: x= 1, print
1 < 8 so do body of loop: not even x, so add 3: x= 4, print
4 < 8 so do body of loop: even x, so decrement x: x= 3, print
3 < 8 so do body of loop: not even x, so add 3: x= 6, print
6 < 8 so do body of loop: even x, so decrement x: x= 5, print
5 < 8 so do body of loop: not even x, so add 3: x= 8, print
not 8 < 8: done with while statement

3: 3 -1 2

| line | n | k | comment |
|---|---|---|---|
| | | | index `0 1 2 3` |
| 1 | | | v value `5 3 -2 1` |
| 2 | 0 | | |
| 3 | | 1 | 1<4 true |
| 4 | 3 | | 0+1*v[1] = v[1] = 3 |
| 5 | | | print 3 |
| 3 | | 2 | 1+1=2; 2<4 true |
| 4 | -1 | | 3+2*v[2] = 3+2*(-2) = -1 |
| 5 | | | print -1 |
| 3 | | 3 | 2+1=3; 3<4 true |
| 4 | 2 | | -1+3*v[3] = -1+3*(1) = 2 |
| 5 | | | print 2 |
| 3 | | 4 | 3+1=4; 4<4 false |

4: `offoffonondowndownupup`
(loop appends two copies of each array element, traversing the array in reverse order)

5:
```
static void ASqr(int[] x)
{
   for (int i = 0; i < x.Length; i++) {
     x[i] = x[i]*x[i];
   }
}

static int AProd(int[] x)
{
   int prod = 1;
   for (int i = 0; i < x.length; i++) {
     prod = prod*x[i];
   }
   return prod;
}
```

6:
```
{ // IndexOf returns -1 when not found
   errors = 0;
   foreach(string s in e) {
     if (s.IndexOf("error")) >= 0) {
        errors++;
     }
   }
   return errors;
 }
```

7:
```
{
   Console.Write(" " + n);
   while (n != 0) {
     n /= k;
     Console.Write(" " + n);
   }
}
```

8:
```
{
   int count = 0;
   for (int i = 0; i < n; i++) {
     if (rand.Next(0, x) < k) {
       count++;
     }
   }
   return count;
```

9:
```
{
   if (n > 0)
     for(int i = 0; i <= n; i+=2) {
       Console.Write(" " + i);
   } else {
     for(int i = 0; i >= n; i-=2) {
       Console.Write(" " + i);
     }
   }
}
```

```
10:{ //can count both yes and no, or:
     int extraYes = 0;
     Console.WriteLine("Enter votes:");
     string ch = Console.ReadLine();
     while (ch != "q") {
       if (ch == "y") {
         extraYes++;
       } else {
         extraYes--;
       }
       ch = Console.ReadLine();
     }
     if (extraYes > 0)
       Console.WriteLine("Yes won");
     else if (extraYes < 0)
       Console.WriteLine("No won");
     else
       Console.WriteLine("A tie");
   }
```

```
11:
while (!reader.EndOffStream) {
   Console.WriteLine(reader.ReadLine());
   reader.ReadLine(); //OK to produce null
}
reader.Close();
```

// OR

```
bool doPrint = true;
while (!reader.EndOffStream) {
   string s = reader.ReadLine();
   if (doPrint) {
      Console.WriteLine(s);
   }
   doPrint = !doPrint;
}
reader.Close();
```

```
12:
{
  int[] counts = new int[chars.Length];
  for (int i = 0; i < chars.Length; i++){
    foreach(char ch in source) {
      if(chars[i]==ch) {
        counts[i]++;
      }
    }
  }
  return counts;
}
```

```
13:  16 6 -4
coming from
2*a[2], a[3], (a[1] - a[0]),
then 2*8 6 (5-9)
```

```
14:
12345678901234567890
  2.20  4.44  8.89
```

first line shows alignment, then field width 6, round each to 2 decimal places

```
15: 4 88

(int)((5/3)*4.0)
= (int)((1)*4.0)   integer quotient
= (int)(4.0)
= 4

(int)((8.88/2)*20)
= (int)(88.8)
= 88 casting to int truncates fractional
part
```

```
16: 1 51 62 52 63 53 6
```

Finish inner loop before returning to outer loop heading:

```
line a b comment
1    1   1<4, do outer loop
2      5 5<7 do inner loop
3        print 1 5 (no blank after 5)
2      6 5+1=6; 6<7 continue inner loop
3        print 1 6
2      7 6+1=7; 7 not < 7; end inner loop
1    2   1+1=2; 2<4, do outer loop
2      5 5<7 do inner loop
3        print 2 5
2      6 5+1=6; 6<7 continue inner loop
3        print 2 6
2      7 6+1=7; 7 not < 7; end inner loop
1    3   2+1=3; 3<4, do outer loop
2      5 5<7 do inner loop
3        print 3 5
2      6 5+1=6; 6<7 continue inner loop
3        print 3 6
2      7 6+1=7; 7 not < 7; end inner loop
1    4   3+1=4; 4 not < 4; end
```

17. If we let the row and column indices r and c start from 0, then the highest value is n below. We go through all row and column positions calculating the minimum distance to an edge.
An example: In the first example n is 4. The character at row 3, column 2 (counting from 0) is 'b', and min distance to edge vertically is min(3, 4-3) = 1; min distance horizontally is min(2, 4-2) = 2; overall min(1, 2) is 1, so the character chosen is s[1]: 'b'. Same idea in general:

```
{
  int n = 2*s.Length - 2;
  for (int r = 0; r <= n; r++) {
    int rmin = Math.Min(r, n-r);
    for (int c = 0; c <= n; c++) {
      int cmin = Math.Min(c, n-c);
      int d = Math.Min(rmin, cmin);
      Console.Write(s[d]);
    }
    Console.WriteLine();
  }
}
```