1. (20pts) Give 3 legal/valid variable names in Java, two of which are NOT solely alphabetic.
   a) (1.5 pt each) answers vary, but something like
      firstName, f_name, and name123
   b) (1.5 pt each) answers vary, but something like
      int, if, and while
   c) (1.5 pt each) answers vary, but something like
      123name, name%, and first name.
   d) (1.5 pt each) answer vary, but examples are:
      _name (starts with underscore or dollar sign)
      NAME (not camel case capitalization)
      firstName (to describe something like a price)

2. (20pts) ASCII Encoding
   Characters (symbols) in the English alphabet are stored inside the computer as numbers. That
   is, each letter/symbol is encoded as a number. The encoding method used to represent the
   alphabet is known as the ASCII code (American Standard Code for Information Interchange).
   The website http://www.ascii-code.com contains ASCII code tables which shows the encoding
   from a decimal number to an English language character.
   a) Give the series of ASCII characters encoded by the following decimal values:
      72 105 32 70 97 108 108 67 83 33
      Hi FallCS!
      (-1 pt per incorrect character)
b) Give the series of decimal values to encode the following ASCII characters:

```
4%_5P aA)V
52 37 95 53 80 32 97 65 41 86
```

(-1 pt per incorrect code)

3. (20pts) For the following statements, state whether the statement is correct or has an error. If there is an error, describe it. If there is no error, give the value stored by the assignment statement. Evaluate each statement with the original values of the variables given below.

(Note: You will be asked to do similar problems on the midterm and you will not have access to a Java compiler. I recommend you write these statements inside a Java program after you have done the homework. You can check your answers --- if you have errors, understand why.)

Original values of the variables:

```
int    i1 = 0,   i2 = 3,   i3 = 7;
double d1 = 2.0, d2 = 3.5, d3 = 6.8;
String s1 = "5.0", s2 = "!", s3 = "123";
```

a)    i2 = i3;  
    7

b)    d1 = s1;  
    error: can't assign String value to double variable.

c)    i1 = i2 + i3;  
    10

d)    d1 = i1 + i3;  
    7.0

e)    d1 = i2 + d2;  
    6.5

f)    i1 = i3 + d1;  
    error: can't assign double value to integer variable.

g)    i3 = i3 % i2 + i1;  
    1

h)    s1 = i1 + i3;  
    error: can't assign integer value to String variable.

i)    s1 = s2 + s3;  
    "!123" (with or without quotation marks).

j)    s1 = i1 + s1;  
    "05.0"

k)    s1 = i2 – i3 + s3;  
    "-4123"

l)    s1 = s3 + i2 – i1;  
    error: can't use – operator with a String and int operand.

m)    s1 = s3 + i1 + d2;  
    "12303.5"

n)    s1 = i1 + d2 + s1;  
    "3.55.0"

o)    i3 = (int) d3;  
    6

p)    d3 = (int) d3 + (int) d2;  
    9.0

q)    d3 = (int) d3 + d2;  
    9.5

r)    s1 = s3 + i2 * i1;  
    "1230"

s)    i3 = i3 * i3 / i2;  
    16

t)    i3 = i3 / i2 * i3;  
    14
public class 1stHW {
    public static <void> Main(String[] args) <{>
        int x = 14;
        System.out.print("The value of x is" + "");
        System.out.println(x);
        int x = 13;
        int y = x + 4<;>
        System.Out.Println(y);
        18 = y;
        x + 4;
        y = 15;
        System.out.println(x + " " + y);
        String s = "Hello";
        System.out.println(s);
        y = x%;
    }
}

a) (10pts) Identify the errors by circling them in the code above.
    errors highlighted in red above. code in <> was added. There were actually 12 errors,
    so identifying any 10 is a full credit answer.
    1 point per error/correction
    -1 per incorrect error/correction

b) (10pts) Rewrite the code so that all the errors are removed or corrected.
    public class HW1 {
        public static void main(String[] args) {
            int x = 14;
            System.out.print("The value of x is");
            System.out.println(x);
            x = 13;
            int y = x + 4;
            System.out.println(y);
            y = 18;
            x = x + 4;
            y = 15;
            System.out.println(x + " " + y);
            String s = "Hello";
            System.out.println(s);
            y = x%y;
        }
    }

    answers can vary on this. There were actually 12 errors, so identifying any 10 is a full
    credit answer.
    1 point per error/correction
    -1 per incorrect error/correction

    You can correct several of the misformed statements in multiple ways. For example
    y = x% could be corrected in lots of ways. As long as it's a syntactically correct
    statement, it's fine.
5. Algorithm tracing – Consider the following algorithm, written in pseudocode:

```plaintext
i = 7398;
d = 0;
s = 0;
c = 0;
as long as i does not equal 0:
    d = i % 10;
i = i / 10;
s = s + d;
c = c + 1;
//complete table entry below
```

a) (15pts) Fill out the table below and trace each step (or loop) of the algorithm above (see comment). The initial state of the variables has been completed for you. Use only as many rows as you need.

<table>
<thead>
<tr>
<th>Step</th>
<th>i</th>
<th>d</th>
<th>s</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>7398</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>739</td>
<td>8</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>73</td>
<td>9</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>3</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>7</td>
<td>27</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b) (5pts) In your own words, explain what the values of s and c are after the algorithm finishes. DO NOT simply restate the algorithm. Think about how the values of s and c relate to the original value of i.

After the algorithm completes, s is the sum of all the digits in the original number 1. So $7 + 3 + 9 + 8 = 27$. c is a count of the number of digits in i. So 7398 has 4 digits. Restating the algorithm is not sufficient. (eg "d is a digit from i and it's added to s and then c increases by 1.")