Joel Edwards
Course: Java Programming 1
Homework 2
April 7, 2011

A.1) "new" is a reserved word.

A.2) This is a problem if "y" has not yet been declared.

A.3) No output as the only statement which would generate output is commented out.

A.4) The results are the same.
`char c = 'A'` - assigns a character constant to 'c'
`char c = 65` - casts the integer value 65 to a character before storing to 'c'.

A.5)
`r3d3` - Valid
`_12345` - Valid
`California` - Valid
`U.S.A` - Invalid because the . is a separator
`$value` - Valid
`Private` - Valid
`2number` - Invalid because identifiers cannot begin with a digit
`X(3)` - Invalid because parenthesis are disallowed within identifier names

A.6)
```
37 % 5 % 3 = 2
37 % (5 % 3) = 1
```

A.7)
```
m = 42
n = 41
```

A.8)
```
m = 6
n = 1
```

A.9)
a. `float f = 345.6;` - it won't compile due to a difference in precision between the variable and the literal
b. `int a = 35;` - No issue

A.10) This will not compile correctly because because there is no guarantee that the variable 'n' has been initialized.

A.11)

a. zero (0)
b. one (1)

A.12)
a = 44 - The value is truncated to the least significant byte

A.13)
```
0
throws ArrayIndexOutOfBoundsException
```

A.14)
```
int i = 1;
while (i <= m) {
    System.out.println(i);
    i++;
}
```

A.15)
a.
g = 25
c = 6
b.
g = 0
c = 5

A.16)
a. x = 15
b. x = 35
c. x = 35

A.17) half = 0

A.18)
```
int totalSeconds = 370;
int minutes = totalSeconds / 60;   // whole minutes
int seconds = totalSeconds % 60;   // remaining seconds
```

A.19) a = 10

B.1)

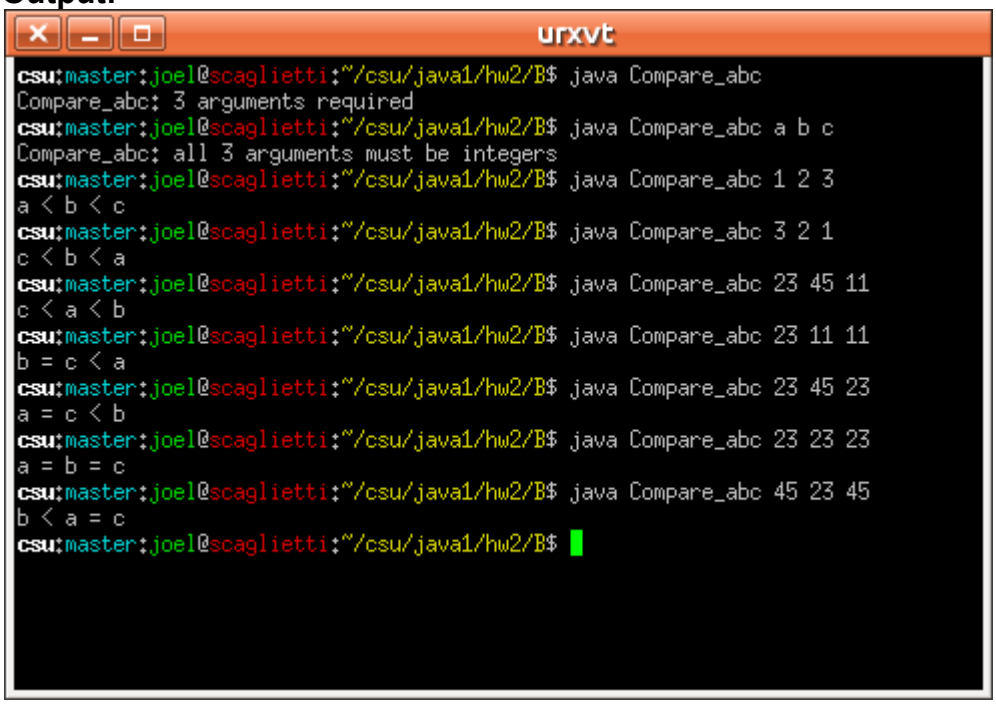
Source:

Compare_abc.java:

```
import java.util.ArrayList;
import java.util.Comparator;
import java.util.Iterator;
import java.util.TreeSet;

class Compare_abc
{
    private static final int ARG_COUNT = 3;

    public static void main(String args[]) {
        if (args.length != ARG_COUNT) {
            System.out.println("Compare_abc: " +ARG_COUNT+ "
arguments required");
            System.exit(1);
        }

        TreeSet<ArrayList<Object>> tree = new
TreeSet<ArrayList<Object>>(new Comparator<ArrayList<Object>>()
        {
            public int compare(ArrayList<Object> a, ArrayList<Object>
b) {
                Integer a_val = (Integer)a.get(1);
                Integer b_val = (Integer)b.get(1);
                if (a_val > b_val) {
                    return 1;
                } else if (a_val < b_val) {
                    return -1;
                } else {
                    String a_name = (String)a.get(0);
                    String b_name = (String)b.get(0);
                    return a_name.compareTo(b_name);
                }
            }
        });

        try {
            for (int i = 0; i < args.length; i++) {
                ArrayList<Object> p = new ArrayList<Object>();
                p.add((char)((65 | 32) + i) + "");
                p.add(Integer.parseInt(args[i]));
                tree.add(p);
            }
```

```
        } catch (NumberFormatException e) {
            System.out.println("Compare_abc: all " +ARG_COUNT+ "
arguments must be integers");
            System.exit(1);
        }

        Iterator iter = tree.iterator();

        ArrayList<Object> last = (ArrayList<Object>)(iter.next());
        String name = (String)last.get(0);
        System.out.print(name);
        while (iter.hasNext()) {
            String separator = "<";
            ArrayList<Object> item = (ArrayList<Object>)
(iter.next());
            Integer val = (Integer)item.get(1);
            Integer last_val = (Integer)last.get(1);
            if (val == last_val) {
                separator = "=";
            }
            name = (String)item.get(0);
            System.out.print(" " +separator+ " " +name);
            last = item;
        }
        System.out.println("");
    }
}
```

Output:



B.2)

Source:

Digits.java:

```
class Digits
{
    public static void main(String args[]) {
        if (args.length != 1) {
            usage();
        }

        int digit = 0;
        try {
            digit = Integer.parseInt(args[0]);
        } catch (NumberFormatException e) {
            usage();
        }

        if ((digit < 0) || (digit > 9)) {
            usage();
        }

        String text = "";

        switch (digit) {
            case 0:  text = "zero";   break;
            case 1:  text = "one";    break;
            case 2:  text = "two";    break;
            case 3:  text = "three";  break;
            case 4:  text = "four";   break;
            case 5:  text = "five";   break;
            case 6:  text = "six";    break;
            case 7:  text = "seven";  break;
            case 8:  text = "eight";  break;
            case 9:  text = "nine";   break;
            //default: text = "uknown"; break;
        }

        System.out.println(text);
    }

    private static void usage() {
        System.out.println("usage: Digits <digit>");
        System.out.println("       digit must be an integer digit
between 0 and 9");
        System.exit(1);
    }
}
```

Output:

```
csu:master:joel@scaglietti:~/csu/java1/hw2/B$ java Digits
usage: DigitName <digit>
       digit must be an integer digit between 0 and 9
csu:master:joel@scaglietti:~/csu/java1/hw2/B$ java Digits 0
zero
csu:master:joel@scaglietti:~/csu/java1/hw2/B$ java Digits 1
one
csu:master:joel@scaglietti:~/csu/java1/hw2/B$ java Digits 2
two
csu:master:joel@scaglietti:~/csu/java1/hw2/B$ java Digits 3
three
csu:master:joel@scaglietti:~/csu/java1/hw2/B$ java Digits 4
four
csu:master:joel@scaglietti:~/csu/java1/hw2/B$ java Digits 5
five
csu:master:joel@scaglietti:~/csu/java1/hw2/B$ java Digits 6
six
csu:master:joel@scaglietti:~/csu/java1/hw2/B$ java Digits 7
seven
csu:master:joel@scaglietti:~/csu/java1/hw2/B$ java Digits 8
eight
csu:master:joel@scaglietti:~/csu/java1/hw2/B$ java Digits 9
nine
csu:master:joel@scaglietti:~/csu/java1/hw2/B$
```

B.3)

Source:

Graph.java:

```
class Graph
{
    public static void main(String args[]) {
        for (String arg: args) {
            try {
                Integer.parseInt(arg);
            } catch (NumberFormatException e) {
                usage();
            }
        }

        for (String arg: args) {
            StringBuilder builder = new StringBuilder();
            int length = Integer.parseInt(arg);
            for (int i=0; i<length; i++) {
                builder.append('*');
            }
            System.out.println(builder.toString());
        }
```

```
    }

    private static void usage() {
        System.out.println("usage: DigitName [val1 [val2 ...]]");
        System.out.println("       val1, val2, etc. must be
integers");
        System.exit(1);
    }
}
```

Output:

B.4)

Source:

BubbleSort.java:

```
class BubbleSort
{
    public static void main(String args[]) {
        double[] values = new double[args.length];

        for (int i = 0; i < args.length; i++) {
            try {
                values[i] = Double.parseDouble(args[i]);
            } catch (NumberFormatException e) {
                usage();
            }
        }

        sort(values);

        for (double value: values) {
            System.out.print(value+ " ");
        }
        System.out.println("");
    }

    private static void sort(double[] values) {
        boolean swap_occurred = true;
        int count = values.length;
        double temp = 0;
        while (swap_occurred) {
            swap_occurred = false;
            for (int i = 0; i < (count - 1); i++) {
                if (values[i] > values[i+1]) {
                    temp = values[i];
                    values[i] = values[i+1];
                    values[i+1] = temp;
                    swap_occurred = true;
                }
            }
            count--;
        }
    }

    private static void usage() {
        System.out.println("usage: DigitName [val1 [val2 ...]]");
        System.out.println("       val1, val2, etc. must be
integers");
        System.exit(1);
```

```
    }
}
```

Output:

```
urxvt
csu:master:joel@scaglietti:~/csu/java1/hw2/B$ make
javac -g -Xlint BubbleSort.java
csu:master:joel@scaglietti:~/csu/java1/hw2/B$ java BubbleSort

csu:master:joel@scaglietti:~/csu/java1/hw2/B$ java BubbleSort 1
1.0
csu:master:joel@scaglietti:~/csu/java1/hw2/B$ java BubbleSort 3 2 1
1.0 2.0 3.0
csu:master:joel@scaglietti:~/csu/java1/hw2/B$ java BubbleSort 5 4 3 2 1
1.0 2.0 3.0 4.0 5.0
csu:master:joel@scaglietti:~/csu/java1/hw2/B$ java BubbleSort 2 4 2 6 7 2  4 5 2
 5 3 6 7 3
2.0 2.0 2.0 2.0 3.0 3.0 4.0 4.0 5.0 5.0 6.0 6.0 7.0 7.0
csu:master:joel@scaglietti:~/csu/java1/hw2/B$ java BubbleSort 3.4 99.5 2.3 75.4
67.4 77.4
2.3 3.4 67.4 75.4 77.4 99.5
csu:master:joel@scaglietti:~/csu/java1/hw2/B$
```

B.5)

Source:

MagicSquares.java:

```
class MagicSquares
{
    public static void main(String args[]) {
        if (args.length != 1) {
            usage();
        }

        int n = 0;
        try {
            n = Integer.parseInt(args[0]);
        } catch (NumberFormatException e) {
            usage();
        }

        if ((n % 2) != 1) {
            usage();
        }
```

```
        int[][] result = solve(n);

        display(result, n);
    }

    public static void display(int[][] square, int n) {
        int cmax;
        int max = cmax = n * n;
        int width = 1;
        while (cmax >= 10) {
            width++;
            cmax /= 10;
        }

        for (int y = 0; y < n; y++) {
            for (int x = 0; x < n; x++) {
                System.out.print(String.format(String.format("  %%
%dd", width), square[x][y]));
            }
            System.out.println("");
        }
    }

    public static int[][] solve(int n) {
        int max = n * n;
        int[][] square = new int[n][n];
        int x = n / 2;
        int y = 0;

        for (int k = 1; k <= max; k++) {
            if (square[x][y] != 0) {
                if (y == (n - 2)) {
                    y = 0;
                } else if (y == (n - 1)) {
                    y = 1;
                } else {
                    y += 2;
                }
                if (x == (n - 1)) {
                    x = 0;
                } else {
                    x++;
                }
            }
            square[x][y] = k;
            if (x == 0) {
                x = n - 1;
            } else {
```

```
                x--;
            }
            if (y == 0) {
                y = n - 1;
            } else {
                y--;
            }
        }

        return square;
    }

    private static void usage() {
        System.out.println("usage: MagicSquares <size>");
        System.out.println("       size - must be an odd integer");
        System.exit(1);
    }
}
```

Output: