I. (10 pts., 2 pts. each) Short answer. Answer the following questions concerning the program below.

(1) What will be the file name for the program?

(2) What variable names are declared in the first method?

(3) What are the names of the methods?

(4) What is the return type of the second method?

(5) What are the types of the parameters in the second method?

```java
public class Vocabulary {
    // first method
    public static void main(String[] args) {
        String text = "Java is exciting!";
        boolean b = hasChar(text, 'x');
        System.out.println(b);
    }

    // second method
    public static boolean hasChar(String s, char c) {
        return (-1 != s.indexOf(c));
    }
}
```
II. (10 pts.) The following program is missing a lot of notation. Fill in each of the boxes in the program. Select from the list below the program. [Note: some are used more than once; some not at all.] The program should print “That is a lucky number!” if the user enters 13 or 42.

```java
public class Notation {
    public static final Scanner CONSOLE = new Scanner(System.in);

    public static void main(String[] args) {
        System.out.print("Enter a number: ");
        int num = CONSOLE.nextInt();
        if (num == 13 || num == 42) {
            System.out.println("That is a lucky number! ");
        }
    }
}
```

! != -= -= && , = == > >= [ \n / \ \
 */ " \\
*/
*== /= \n++ < {  
+= <= ||
III. (10 pts.) What does the following program print out? Mark your answer clearly.

```java
public class MysteryProgram {
    public static void main(String[] args) {
        int a = 4;
        a = a + 1;
        System.out.print(a);
        String s = "uppercase";
        System.out.print(s.substring(0, 2));
        System.out.print(a - 2);
        String t = "AFFIRM";
        t = t.toLowerCase();
        System.out.print(t.substring(t.length() - 2, t.length()));
        System.out.print(Math.min(a, 4));
        System.out.println("n");
    }
}
```
IV. (10 pts.) What does the following program print out? Mark your answer clearly.

```java
public class MysteryMethod {
    public static void main(String[] args) {
        int[] a = {-1, 0, 2, 4, 6};
        mystery(a);
        for (int i = 0; i < a.length; i++) {
            System.out.print(a[i] + " ");
        }
        System.out.println();
    }

    public static void mystery(int[] list) {
        for (int i = 1; i < list.length; i++) {
            list[i] = 2 * list[i] + 1;
        }
    }
}
```
V. (10 pts.) Write a static method named `minimum` that takes in three integer parameters, named `a`, `b`, and `c`, which returns an integer representing the minimum value of `a`, `b`, and `c`. You may use `Math.min` if you wish.
VI. (10 pts.) For each of the following code segments, clearly indicate what the code will print:

```java
// 3 pts.
int index = 1;
while (index < 10) {
    System.out.print(index + ";");
    index = index * 2;
}
```

```java
// 3 pts.
String s = "12345";
System.out.println(s.length());
System.out.println(s.charAt(1));
System.out.println(s + ";" + s);
```

```java
// 4 pts.
public static void main(String[] args){
    int index = 1;
    while (index <= 6) {
        int result = simpleMethod(index);
        System.out.print(result + ";");
        index++;
    }
}

public static int simpleMethod(int n) {
    if (n % 2 == 0) {
        return n;
    } else {
        return n * n;
    }
}
```
VII. (10 pts., 2 pts. each) Consider the following method:

```java
public static int mystery(int k) {
    if (k <= 0) {
        k = k * 2;
        return k;
    } else {
        k = k + 4;
    }
    if (k % 2 == 0) {
        k = k + k % 10;
    }
    return k;
}
```

Give the output of calling the method with the values below.

- mystery(-3)
- mystery(0)
- mystery(5)
- mystery(10)
- mystery(18)
VIII. (10 pts.) Write a for-loop to print out the squares of the numbers from 6 to 1. That is, your program should print out

36, 25, 16, 9, 4, 1

with all numbers on the same line and separated by commas.
IX. (10 pts.) Write a static method named `printSumOfTerms` that takes in an array of integers, named `vals`. Your `printSumOfTerms` method should print to `System.out` all of the numbers in the array separated by " + " (that is, a space, the plus sign, and another space), then an equals sign, and then their sum. For example, the following two calls to `printSumOfTerms`:

```java
int[] a = {1, 4, 2};
int[] b = {27};
printSumOfTerms(a);
printSumOfTerms(b);
```

should output:

```
1 + 4 + 2 = 7
27 = 27
```
X. (10 pts.) Write a method named `getOneToTen` that has one parameter: `prompt` (type `String` class). `prompt` should be printed whenever the user needs to enter a token.

The method should return an `int` value between 1 and 10 read in from the keyboard. A while loop that includes a lookahead method (a `Scanner` method that tests values before they are read) should be used to handle possible user errors: the user might enter a value less than 1, the user might enter a value larger than 10, or the user might enter a token that is not an `int`. You can assume that `java.util.*` has been imported and that `CONSOLE` is a `Scanner` class constant for reading from the keyboard.
### Table 3.2 Useful Static Methods in the Math Class

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs</td>
<td>absolute value</td>
<td>Math.abs(-308) returns 308</td>
</tr>
<tr>
<td>ceil</td>
<td>ceiling (rounds upward)</td>
<td>Math.ceil(2.13) returns 3.0</td>
</tr>
<tr>
<td>cos</td>
<td>cosine (radians)</td>
<td>Math.cos(Math.PI) returns -1.0</td>
</tr>
<tr>
<td>exp</td>
<td>exponent base e</td>
<td>Math.exp(1) returns 2.7182818284590455</td>
</tr>
<tr>
<td>floor</td>
<td>floor (rounds downward)</td>
<td>Math.floor(2.93) returns 2.0</td>
</tr>
<tr>
<td>log</td>
<td>logarithm base e</td>
<td>Math.log(Math.E) returns 1.0</td>
</tr>
<tr>
<td>log10</td>
<td>logarithm base 10</td>
<td>Math.log10(1000) returns 3.0</td>
</tr>
<tr>
<td>max</td>
<td>maximum of two values</td>
<td>Math.max(45, 207) returns 207</td>
</tr>
<tr>
<td>min</td>
<td>minimum of two values</td>
<td>Math.min(3.8, 2.75) returns 2.75</td>
</tr>
<tr>
<td>pow</td>
<td>power (general exponentiation)</td>
<td>Math.pow(3, 4) returns 81.0</td>
</tr>
<tr>
<td>random</td>
<td>random value</td>
<td>Math.random() returns a random double value k such that 0.0 ≤ k &lt; 1.0</td>
</tr>
<tr>
<td>round</td>
<td>round real number to nearest integer</td>
<td>Math.round(2.718) returns 3</td>
</tr>
<tr>
<td>sin</td>
<td>sine (radians)</td>
<td>Math.sin(0) returns 0.0</td>
</tr>
<tr>
<td>sqrt</td>
<td>square root</td>
<td>Math.sqrt(2) returns 1.4142135623730951</td>
</tr>
<tr>
<td>toDegrees</td>
<td>converts from radians to degrees</td>
<td>Math.toDegrees(Math.PI) returns 180.0</td>
</tr>
<tr>
<td>toRadians</td>
<td>converts from degrees to radians</td>
<td>Math.toRadians(270.0) returns 4.71238898038469</td>
</tr>
</tbody>
</table>

### Table 3.3 Useful Methods of String Objects

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Example (assuming s is &quot;hello&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>charAt(index)</td>
<td>character at a specific index</td>
<td>s.charAt(1) returns 'e'</td>
</tr>
<tr>
<td>endsWith(text)</td>
<td>whether or not the string ends with some text</td>
<td>s.endsWith(&quot;llo&quot;) returns true</td>
</tr>
<tr>
<td>indexOf(text)</td>
<td>index of a particular character or String (-1 if not present)</td>
<td>s.indexOf(&quot;o&quot;) returns 4</td>
</tr>
<tr>
<td>length()</td>
<td>number of characters in the string</td>
<td>s.length() returns 5</td>
</tr>
<tr>
<td>startsWith(text)</td>
<td>whether or not the string starts with some text</td>
<td>s.startsWith(&quot;hi&quot;) returns false</td>
</tr>
<tr>
<td>substring(start, stop)</td>
<td>characters from start index to just before stop index</td>
<td>s.substring(1, 3) returns &quot;el&quot;</td>
</tr>
<tr>
<td>toLowerCase()</td>
<td>a new string with all lowercase letters</td>
<td>s.toLowerCase() returns &quot;hello&quot;</td>
</tr>
<tr>
<td>toUpperCase()</td>
<td>a new string with all uppercase letters</td>
<td>s.toUpperCase() returns &quot;HELLO&quot;</td>
</tr>
</tbody>
</table>
## Useful Methods of Scanner Objects

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>next()</code></td>
<td>Reads and returns the next token as a String</td>
</tr>
<tr>
<td><code>nextDouble()</code></td>
<td>Reads and returns a double value</td>
</tr>
<tr>
<td><code>nextInt()</code></td>
<td>Reads and returns an int value</td>
</tr>
<tr>
<td><code>nextLine()</code></td>
<td>Reads and returns the next line of input as a String</td>
</tr>
<tr>
<td><code>hasNext()</code></td>
<td>Returns true if there is another token to be read</td>
</tr>
<tr>
<td><code>hasNextDouble()</code></td>
<td>Returns true if there is another token to be read and if it can be</td>
</tr>
<tr>
<td></td>
<td>interpreted as a double</td>
</tr>
<tr>
<td><code>hasNextInt()</code></td>
<td>Returns true if there is another token to be read and if it can be</td>
</tr>
<tr>
<td></td>
<td>interpreted as an int</td>
</tr>
<tr>
<td><code>hasNextLine()</code></td>
<td>Returns true if there is another line of input to be read</td>
</tr>
</tbody>
</table>