There are seven questions on this exam, each worth 2.5 points. There is one question per lecture, numbered in order of the lectures, not in order of difficulty. If a question seems difficult to you, skip it and come back to it. You will have 50 minutes to complete the exam.

**This exam is preprocessed by computer.** You answer questions by filling in circles completely with a dark pencil. If you change your mind, you must erase completely and fill in another circle!

Do this ● not this ✔ or this ✗ or this ❌.

**Resources.** You may reference your optional one-sided 8.5-by-11 handwritten "cheat sheet" during this exam. You may not use the textbook, your notes, or any electronic devices. You may not communicate with anyone except the course staff during this exam.

**Discussing this exam.** Due to travel for extracurriculars and sports, some of your peers will take this exam later. Do not discuss its contents with anyone who has not taken it.

**This page.** Do not remove this exam from the exam room. Fill in this page now, but do not start the exam until you are told.

Name

NetID

Precept

Exam Room

“I pledge my honor that I have not violated the Honor Code during this examination.”

___________________________________________________________________
___________________________________________________________________

[copy the pledge here]

_____________________________ [signature]
Q1. Types and Casts (2.5 points).

Suppose that the code at left in each row is the sole argument of a `StdOut.println()` statement. Fill in the circle corresponding to the string it would print.
Q2. Basic definitions (2.5 points).

In the box to the left of each description, write the letter corresponding to the Java language entity that best matches. A letter may be used once, more than once, or not at all, but there is only one correct answer for each box.

- **D** Java data type that has only two values
- **F** A `double` operator
- **F** A `String` operator
- **C** A `double` literal
- **I** Source-code representation of $\pi$
- **B** A set of values and operations on those values
- **E** A Java library function
- **A** Data structure for a sequence of values of the same type
- **I** Java data type for integers
- **I** A `boolean` operator
Q3. Loops and conditionals (2.5 points).

Suppose that this code is in the file Q3.java

```java
public class Q3
{
    public static void main(String[] args)
    {
        int N = Integer.parseInt(args[0]);
        String r = "++";
        for (int i = 1; i <= N; i++)
            if ((i % 2) == 0)
                r = r + i + r;
            else r = i + r + i;
        System.out.println(r);
    }
}
```

and that you compile it using the command `javac-intros Q3`. Study the code to verify that the command `java-intros Q3` 1 prints the string "1+1" and `javac-intros Q3` 2 prints the string "1+2+1+1".

Then fill in the circle on each row corresponding to the specified character. You must fill in exactly one circle in each row.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>first character printed by java-intros Q3</td>
<td></td>
<td></td>
<td></td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>first character printed by java-intros Q3</td>
<td></td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>tenth character printed by java-intros Q3</td>
<td></td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>second character printed by java-intros Q3</td>
<td></td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>next-to-last character printed by java-intros Q3</td>
<td></td>
<td></td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>
### Q4. Arrays (2.5 points).

For each code snippet at left, fill in the one circle at right that describes whether it is legal or whether it will lead to an error.

<table>
<thead>
<tr>
<th>Code</th>
<th>Legal code</th>
<th>Compile-time error</th>
<th>Run-time error</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int[] a;</code></td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>int[] b = 0;</code></td>
<td></td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td><code>int[] c = int[10];</code></td>
<td></td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td><code>int[] d = new int[10];</code></td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>int[][] e = new int[10];</code></td>
<td></td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td><code>int[10] f = new int[10];</code></td>
<td></td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td><code>int[] g = new int g[10];</code></td>
<td></td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td><code>int[] h = {1, 2, 3}; int t = h;</code></td>
<td></td>
<td>☐</td>
<td>☐</td>
</tr>
<tr>
<td><code>int[] i = {1, 2, 3}; int[] u = i;</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>int[] j = {1, 2, 3}; int v = j[3];</code></td>
<td></td>
<td></td>
<td>☐</td>
</tr>
</tbody>
</table>
Q5. I/O (2.5 points).

Fortunately study the following Java program.

```java
public class Q5
{
    public static void main(String[] args)
    {
        int N = args.length;
        String[] a = new String[N * 2];
        for (int i = 0; i < N; i++)
        {
            a[i] = args[i];
            a[i + N] = args[N - i - 1];
        }
        for (int i = 0; i < a.length; i++)
            StdOut.print(a[i] + " ");
        StdOut.println();
    }
}
```

In the box to the left of each command, write the letter corresponding to the output it produces. A letter may be used once, more than once, or not at all.

```
E  java-introcs Q5 now is
A  is is is is
B  is is now now
A  java-introcs Q5 is is
C  is now is now
D  java-introcs Q5 is now
D  is now now is
E  java-introcs Q5 is now | java-introcs Q5 now is
E  now is is now
F  now is now is
G  now now is is
H  java-introcs Q5 is is | java-introcs Q5 now now
H  now now now now
```
Q6. Methods (2.5 points).

Consider the following Java program, which has no purpose other than to test your understanding of methods, arguments, and scope.

```java
public class Q6 {
    public static void f(int[] a, int k, int m) {
        double[] b = f(a, 2.6);
        a[0] = a[0] * m;
        a[2] = (int) (b[0] + b[1]);
        m = m + 5;
    }
    public static double[] f(int[] a, double x) {
        double[] b = new double[2];
        b[0] = a[0] + x;
        b[1] = a[1] + x;
        return b;
    }
    public static void main(String[] args) {
        int[] a = {2, 3, 4};
        int k = 5;
        int m = 5;
        f(a, k, m);
    }
}
```

Fill in the one circle on each row corresponding to the value of the given expression just after the call `f(a, k, m)` has returned control to `main()`.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>⬜</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>⬜</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a[0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>⬜</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a[1]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>⬜</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a[2]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>⬜</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Q7. Recursion (2.5 points).

Consider the following recursive function:

```java
public static int ack(int m, int n)
{
    if (m == 0) return n + 1;
    if (n == 0) return ack(m - 1, 1);
    return ack(m - 1, ack(m, n - 1));
}
```

Fill in the circle corresponding to the value of `ack()` for each given value of the arguments. You must fill in exactly one circle in each row.