This exam has 8 questions (including question 0) worth a total of 70 points. You have 50 minutes. Write all answers inside the designated spaces.

**Policies.** The exam is closed book, except that you are allowed to use a one page cheatsheet (8.5-by-11 paper, one side, in your own handwriting). No electronic devices are permitted.

**Discussing this exam.** Discussing the contents of this exam before solutions have been posted is a violation of the Honor Code.

**This exam.** Do not remove this exam from this room. Print your name, NetID, and the room in which you are taking the exam in the space below. Mark your precept number. Also, write and sign the Honor Code pledge. You may fill in this information now.

Name:

NetID:

Precept:  

Exam room:

“I pledge my honor that I will not violate the Honor Code during this examination.”

Signature

<table>
<thead>
<tr>
<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>total</th>
</tr>
</thead>
</table>


0. Miscellaneous. (2 points)

(a) Write your name, NetID, and the room in which you are taking the exam in the space provided on the front of this exam.
(b) Mark your precept number on the front of this exam.
(c) Write and sign the Honor Code pledge on the front of this exam.

1. Java basics. (10 points)

Give the value of each of the following Java expressions. To express your answer, write a Java literal of the appropriate type, such as 0, 0.0, false, or "0". If an expression results in a compile-time or run-time error, write ERROR for its value. Assume that the variables x, y, and z have been initialized as follows:

```java
int x = 1;
int y = 2;
int z = 3;
```

<table>
<thead>
<tr>
<th>Java expression</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>1.0 * x + y / z</td>
<td></td>
</tr>
<tr>
<td>((x &gt; y)</td>
<td></td>
</tr>
<tr>
<td>Math.min(Math.max(x, y), y*y % z)</td>
<td></td>
</tr>
<tr>
<td>Math.sqrt(x - y - z)</td>
<td></td>
</tr>
<tr>
<td>Double.parseDouble(x + &quot;2&quot; + y)</td>
<td></td>
</tr>
</tbody>
</table>
2. Properties of arrays and functions. (10 points)

(a) Which of the following statements are true for Java arrays? Mark each statement as either true or false.

- An array can’t simultaneously store both an element of type double and an element of type boolean.
- Once you create an array, you cannot change its type.
- You must access the elements in an array in sequential order, e.g., you cannot access a[5] until you have accessed a[0], a[1], through a[4].
- If a[] is a boolean array of length 126, then the expression a[i] will evaluate arbitrarily to either true or false if the index i is equal to 126.
- If a[] and b[] are two arrays of length 2, then a == b is true if and only if both a[0] == b[0] and a[1] == b[1] are true.

(b) Which of the following statements are true for Java functions (static methods). Mark each statement as either true or false.

- A function can accept more than one argument.
- You can use double[] as the return type of a function.
- A function can call other functions only if those functions are defined in the same .java file.
- Two functions defined in the same .java file can have the same name only if they have a different number of arguments.
- If you pass a (reference to an) array as an argument to a function, the function can not only read the array’s entries but it can also change them.
3. Debugging and arrays. (10 points)

Suppose that a[] is an integer array of length n. Consider this incomplete Java code:

```java
// SUBSTITUTE LOOP FRAGMENT HERE
{
    int temp = a[i];
    a[i] = a[n-i-1];
    a[n-i-1] = temp;
}
```

Suppose that you substitute each loop fragment below into the the rectangle above. What effect will it have on the elements in the array a[]?

For each loop fragment at left, write the letter of the best-matching description from the right. You may use each letter once, more than once, or not at all.

- for (int i = 0; i < n; i++)  A. puts elements in reverse order
- for (int i = 0; i < n; i++)  B. puts elements in reverse order when n is even
- for (int i = 0; i < n/2; i++)  C. puts elements in reverse order when n is odd
- for (int i = 0; i <= n; i++)  D. keeps elements in original order
- for (int i = 0; i < n/2; i++)  E. infinite loop
- for (int i = 0; i <= n/2; i++)  F. compile-time error
- for (int i = 0; i < n/2; i++)  G. run-time error

- for (int i = 0; i <= n/2; i++)
4. Functions. (10 points)

The `oddParity()` function takes three boolean arguments and returns `true` if an odd number (either 1 or 3) of its arguments are `true`; otherwise, it returns `false`. Complete two implementations of `oddParity()` by filling in the letter of one of the expressions below in each provided space. You may use a letter once, more than once, or not at all. No other code is allowed.

A. `true`
B. `false`
C. `x`
D. `y`
E. `z`
F. `!z`
G. `x && y`
H. `x || y`
I. `x && y && z`
J. `x || y || z`
K. `count++`

(a)

```java
public static boolean oddParity(boolean x, boolean y, boolean z) {
    int count = 0;
    if (_______) ______;
    if (_______) ______;
    if (_______) ______;
    return (count % 2) != 0;
}
```

(b)

```java
public static boolean oddParity(boolean x, boolean y, boolean z) {
    if (_______) return ______;
    else if (_______) return ______;
    else return ______;
}
```
5. Recursion. (10 points)

Consider the following recursive function:

```java
public static String f(int n) {
    if (n <= 0) return "0";  // statement 1
    if (n == 1) return "1";  // statement 2
    String first = f(n-1);  // statement 3
    String second = f(n-2); // statement 4
    String third = f(n-1);  // statement 5
    return first + n + second + third;  // statement 6
}
```

(a) Complete the following table by filling in the values of $f(2)$ and $f(3)$.

<table>
<thead>
<tr>
<th>n</th>
<th>f(n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>&quot;0&quot;</td>
</tr>
<tr>
<td>1</td>
<td>&quot;1&quot;</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

(b) Mark each of the following statements as either true or false.

<table>
<thead>
<tr>
<th>true</th>
<th>false</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>f(5) returns a string of length $\geq 60$.</td>
</tr>
</tbody>
</table>

Suppose that statements 2 and 3 are exchanged. Then, the modified function returns the same value as the original function for every integer argument $n$.

Instead, suppose that statement 5 is removed and statement 6 is replaced with

```
return first + n + second + first;
```

Then, the modified function returns the same value as the original function for every integer argument $n$. 
6. Powers of 2. (8 points)

For each description on the left, write the letter of the best-matching power of 2 from the right. You may use each letter once, more than once, or not at all.

<table>
<thead>
<tr>
<th>Description</th>
<th>Letter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of points a student can earn on Question 6</td>
<td>A. (2^0)</td>
</tr>
<tr>
<td>Number of 1s in the binary representation of (2^{32} + 1)</td>
<td>B. (2^1)</td>
</tr>
<tr>
<td>Number of 1s in the binary representation of (2^{32} - 1)</td>
<td>C. (2^2)</td>
</tr>
<tr>
<td>Number of 1s in the binary representation of (C05126_{16})</td>
<td>D. (2^3)</td>
</tr>
<tr>
<td>Number of 0s (not 1s) in the 16-bit two’s complement representation of (-84_{10})</td>
<td>E. (2^4)</td>
</tr>
<tr>
<td>Smallest positive integer not representable as a Java int.</td>
<td>F. (2^5)</td>
</tr>
<tr>
<td>Number of bits in a TOY instruction</td>
<td>G. (2^6)</td>
</tr>
<tr>
<td>Number of registers in TOY</td>
<td>H. (2^8)</td>
</tr>
<tr>
<td></td>
<td>I. (2^{10})</td>
</tr>
<tr>
<td></td>
<td>J. (2^{12})</td>
</tr>
<tr>
<td></td>
<td>K. (2^{14})</td>
</tr>
<tr>
<td></td>
<td>L. (2^{15})</td>
</tr>
<tr>
<td></td>
<td>M. (2^{16})</td>
</tr>
<tr>
<td></td>
<td>N. (2^{31})</td>
</tr>
<tr>
<td></td>
<td>O. (2^{32})</td>
</tr>
<tr>
<td></td>
<td>P. (2^{64})</td>
</tr>
</tbody>
</table>
7. TOY. (10 points)

Suppose that you load the following into memory locations 10–17 of TOY, set the PC to 10, and press the RUN button.

```
00: ?? ??
01: ?? ??
10: 8A00   R[A] <- M[00]
11: 7101   R[1] <- 1
13: D216   if (R[2] > 0) PC <- 16
14: 1111
15: C012
16: 9101   M[01] <- R[1]
17: 0000   halt
```

For each initial value of memory location 00 given below, write which value appears in memory location 01 upon termination of the TOY program. Specify each value using 4 hex digits.

(a) 00: 0 0 0 1
(b) 00: 0 0 0 6
(c) 00: 0 0 0 9
(d) 00: 1 E A F
(e) 00: F A C E
TOY REFERENCE CARD

INSTRUCTION FORMATS

| . . . . | . . . . | . . . . | . . . . |
Format RR:  | opcode | d | s | t | (1-6, A-B)
Format A:   | opcode | d | addr | (7-9, C-F)

ARITHMETIC and LOGICAL operations
1: add \(R[d] \leftarrow R[s] + R[t]\)
2: subtract \(R[d] \leftarrow R[s] - R[t]\)
3: and \(R[d] \leftarrow R[s] \& R[t]\)
4: xor \(R[d] \leftarrow R[s] \oplus R[t]\)
5: shift left \(R[d] \leftarrow R[s] \ll R[t]\)
6: shift right \(R[d] \leftarrow R[s] \gg R[t]\)

TRANSFER between registers and memory
7: load address \(R[d] \leftarrow \text{addr}\)
8: load \(R[d] \leftarrow M[\text{addr}]\)
9: store \(M[\text{addr}] \leftarrow R[d]\)
A: load indirect \(R[d] \leftarrow M[R[t]]\)
B: store indirect \(M[R[t]] \leftarrow R[d]\)

CONTROL
0: halt \(\text{halt}\)
C: branch zero if \((R[d] == 0)\) \(PC \leftarrow \text{addr}\)
D: branch positive if \((R[d] > 0)\) \(PC \leftarrow \text{addr}\)
E: jump register \(PC \leftarrow R[d]\)
F: jump and link \(R[d] \leftarrow PC; PC \leftarrow \text{addr}\)

Register 0 always reads 0.
Loads from \(M[\text{FF}]\) come from stdin.
Stores to \(M[\text{FF}]\) go to stdout.

16-bit registers (using two’s complement arithmetic)
16-bit memory locations
8-bit program counter
This page is provided as scratch paper. If you tear it out, please write your name, NetID, and precept number in the space provided and return it inside your exam.

Name: ______________________  NetID: ________  Precept: ______