This test has 8 questions, weighted as indicated. The exam is closed book, except that you are allowed to use a one-page single-sided cheatsheet. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided.

*Print your name, login ID, and precept number on this page* (now), and write out and sign the Honor Code pledge before turning in this paper. It is a violation of the Honor Code to discuss this exam until everyone in the class has taken the exam. You have 50 minutes to complete the test.

**Write out and sign the Honor Code pledge before turning in the test:**

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

Pledge: ____________________________________________

Signature: __________________________________________

Name: ____________________________________________

NetID: ____________________________________________

Precept: ____________________________________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/12</td>
</tr>
<tr>
<td>2</td>
<td>/10</td>
</tr>
<tr>
<td>3</td>
<td>/9</td>
</tr>
<tr>
<td>4</td>
<td>/9</td>
</tr>
<tr>
<td>5</td>
<td>/8</td>
</tr>
<tr>
<td>6</td>
<td>/9</td>
</tr>
<tr>
<td>7</td>
<td>/6</td>
</tr>
<tr>
<td>8</td>
<td>/7</td>
</tr>
<tr>
<td>Total</td>
<td>/70</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Precept</th>
<th>Time</th>
<th>Instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>P01</td>
<td>12:30 TTh</td>
<td>Dave Pritchard</td>
</tr>
<tr>
<td>P01A</td>
<td>12:30 TTh</td>
<td>Donna Gabai</td>
</tr>
<tr>
<td>P01B</td>
<td>12:30 TTh</td>
<td>Pawel Przytycki</td>
</tr>
<tr>
<td>P02</td>
<td>1:30 TTh</td>
<td>Tom Funkhouser</td>
</tr>
<tr>
<td>P02A</td>
<td>1:30 TTh</td>
<td>Allison Chaney</td>
</tr>
<tr>
<td>P02B</td>
<td>1:30 TTh</td>
<td>Pawel Przytycki</td>
</tr>
<tr>
<td>P02C</td>
<td>1:30 TTh</td>
<td>Vivek Pai</td>
</tr>
<tr>
<td>P02D</td>
<td>1:30 TTh</td>
<td>Siddhartha Chaudhuri</td>
</tr>
<tr>
<td>P03</td>
<td>2:30 TTh</td>
<td>Tom Funkhouser</td>
</tr>
<tr>
<td>P03A</td>
<td>2:30 TTh</td>
<td>Allison Chaney</td>
</tr>
<tr>
<td>P04</td>
<td>3:30 TTh</td>
<td>Vivek Pai</td>
</tr>
<tr>
<td>P04B</td>
<td>3:30 TTh</td>
<td>Shilpa Nadimpalli</td>
</tr>
<tr>
<td>P05</td>
<td>7:30 TTh</td>
<td>Shilpa Nadimpalli</td>
</tr>
<tr>
<td>P06</td>
<td>10am WF</td>
<td>Lennart Beringer</td>
</tr>
<tr>
<td>P07</td>
<td>1:30 WF</td>
<td>Dave Pritchard</td>
</tr>
<tr>
<td>P07A</td>
<td>1:30 WF</td>
<td>Kevin Lee</td>
</tr>
<tr>
<td>P07B</td>
<td>1:30 WF</td>
<td>Siyu Liu</td>
</tr>
<tr>
<td>P08</td>
<td>12:30 WF</td>
<td>Donna Gabai</td>
</tr>
<tr>
<td>P08A</td>
<td>12:30 WF</td>
<td>Judi Israel</td>
</tr>
<tr>
<td>P09</td>
<td>11am WF</td>
<td>Judi Israel</td>
</tr>
</tbody>
</table>
1. **Java Expressions** (12 points)

For each of the Java expressions below, write down the type of the expression and its value. If the expression causes a syntax or run-time error, write an X in both boxes.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 + 2 + &quot;3&quot; + 4 + 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(double)(1 / 2 + 1.0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>false &amp;&amp; (!(!true</td>
<td></td>
<td>(true</td>
</tr>
<tr>
<td>7 = 11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>true != false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Double.parseDouble(&quot;1E1&quot;)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2. **Number Systems** (10 points)

For this problem, we ask you to perform several calculations on hexadecimal numbers. For each part, we are using a **16-bit twos-complement** representation.

(a) What is \(0\text{ABE}\), expressed in binary?

(b) What is \(\text{FFEE}\), expressed in decimal?

(c) What is \(\text{F00D} \ ^{\land} \ \text{FEED}\), expressed in hexadecimal?

(d) What is \(\text{B0D1} \ & \ \text{FACE}\), expressed in hexadecimal?

(e) What is \(\text{B0D1} \mid \ \text{FACE}\), expressed in hexadecimal?
3. **Debugging** (9 points)

Recall that the absolute value function of $x$ is defined by

$$\text{abs}(x) = \begin{cases} 
  x, & \text{if } x \geq 0; \\
  -x, & \text{otherwise}.
\end{cases}$$

For example, $\text{abs}(4) = 4$ and $\text{abs}(-2) = 2$.

The following program is supposed to compute the sum of the absolute values of its arguments. Here is a sample run and the expected output:

```
% java AbsoluteSum 1 -2 4
The absolute sum is 7
```

However, your `AbsoluteSum` program is not working. Here is its source code:

```java
public class AbsoluteSum {
    public static void main(String[] args) {
        int n = args.length;
        int sum = 0;
        for (int i = 0; i < n; n++) {
            int value = Integer.parseInt(args(i));
            if (value < 0);
                value = -1 * value;
            sum = sum + value;
        }
        System.out.println("The absolute sum is " + sum);
    }
}
```

For the three parts below, give the line number where there is a bug in the program, and a brief description of the bug. You do not need to write code to fix the bug.

(a) Find a syntax error that prevents the code from compiling.

Line: _______  Description: ________________________________

(b) Find an error that causes the code to loop incorrectly (assuming the previous error was fixed).

Line: _______  Description: ________________________________

After fixing these two bugs, you run the program and find it is computing the wrong value:

```
% java AbsoluteSum 1 -2 4
The absolute sum is -3
```

(c) Find the error that causes this incorrect output.

Line: _______  Description: ________________________________
4. **Arrays** (9 points)

For this problem, you will trace the values stored in three arrays by the following program.

```
public class ThreeArrays {
    public static void main(String[] args) {
        int n = args.length;

        int[] a = new int[n];
        int[] b = new int[n+1];
        int[] c = b;

        for (int i = 0; i < n; i++)
            a[n-i-1] = Integer.parseInt(args[n-i-1]);

        for (int i = 0; i < n; i++)
            b[i+1] = b[i] + a[i];

        for (int i = 0; i < n; i++)
            c[i+1] = b[i] + c[i+1];
    }
}
```

If we run

```
% java ThreeArrays 1 10 100
```

what are the values stored in the arrays at the **end** of the program? Enter your responses in the boxes below.

<table>
<thead>
<tr>
<th>a[0]:</th>
<th>a[1]:</th>
<th>a[2]:</th>
</tr>
</thead>
<tbody>
<tr>
<td>b[0]:</td>
<td>b[1]:</td>
<td>b[2]:</td>
</tr>
<tr>
<td>c[0]:</td>
<td>c[1]:</td>
<td>c[2]:</td>
</tr>
</tbody>
</table>
TOY Reference Card  Use this for the next problem on the facing page.

TOY REFERENCE CARD

INSTRUCTION FORMATS

| . . . . | . . . . | . . . . | . . . . |
Format 1:  | opcode | d | s | t | (0-6, A-B)
Format 2:  | 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 \( R[d] \leftarrow R[s] ^ R[t] \)
5: shift left \( R[d] \leftarrow R[s] << R[t] \)
6: shift right \( R[d] \leftarrow R[s] >> R[t] \)

TRANSFER between registers and memory

7: load address \( R[d] \leftarrow addr \)
8: load \( R[d] \leftarrow \text{mem}[addr] \)
9: store \( \text{mem}[addr] \leftarrow R[d] \)
A: load indirect \( R[d] \leftarrow \text{mem}[R[t]] \)
B: store indirect \( \text{mem}[R[t]] \leftarrow R[d] \)

CONTROL

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

Register 0 always reads 0.
Loads from \( \text{mem}[FF] \) come from stdin.
Stores to \( \text{mem}[FF] \) go to stdout.
\( pc \) starts at 10

16-bit registers
16-bit memory locations
8-bit program counter
A NOOP (no operation) in a TOY program is a command that has no effect, other than that the program counter advances just past this command. One use of NOOPs is as a quick alternative to renumbering all of the lines in your TOY program, when you want to delete a line in the middle.

When we call a command a NOOP, we cannot make any assumptions about the state of the machine. For example, the command 1BB0 is a NOOP since it adds zero to register B, which cannot possibly have any effect on any register or memory location. But the command 1BBA is not a NOOP because, depending on the contents of register A, this might change the value of register B.

Similarly, a pair of commands at memory locations $L$ and $L + 1$ forms a NOOP if reaching line $L$ means that we are guaranteed to get to line $L + 2$, with everything the same as it was at line $L$ (except the program counter).

Determine which of the commands and pairs below are NOOPs. The :: symbols represent hidden parts of the program. Do not make any assumptions about the hidden parts or the initial state of the machine. Circle your YES/NO answer for each of the 8 possible NOOPs.

Use the TOY reference card on the facing page.

<table>
<thead>
<tr>
<th>Line</th>
<th>Command</th>
<th>NOOP?</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>D0D0</td>
<td>YES</td>
</tr>
<tr>
<td>30</td>
<td>BEEF</td>
<td>YES</td>
</tr>
<tr>
<td>40</td>
<td>6991</td>
<td>YES</td>
</tr>
<tr>
<td>41</td>
<td>5991</td>
<td>YES</td>
</tr>
<tr>
<td>50</td>
<td>433E</td>
<td>YES</td>
</tr>
<tr>
<td>51</td>
<td>43E3</td>
<td>YES</td>
</tr>
<tr>
<td>60</td>
<td>2222</td>
<td>YES</td>
</tr>
<tr>
<td>70</td>
<td>3333</td>
<td>YES</td>
</tr>
<tr>
<td>80</td>
<td>DA82</td>
<td>YES</td>
</tr>
<tr>
<td>81</td>
<td>CB82</td>
<td>YES</td>
</tr>
<tr>
<td>90</td>
<td>DA90</td>
<td>YES</td>
</tr>
<tr>
<td>91</td>
<td>CA91</td>
<td>YES</td>
</tr>
</tbody>
</table>
6. **Methods and Input/Output** (9 points)

In this problem, you will analyze the program below:

```java
public class Methodical {
    public static int transform(int x, int y) {
        x = x + 2;
        return (x + y);
    }

    public static int transform(double z) {
        int y = (int) z;
        StdOut.println(y);
        z = z + 1;
        return (int) z;
    }

    public static void main(String[] args) {
        String w = args[0];
        int x = Integer.parseInt(StdIn.readString());
        int y = Integer.parseInt(args[1]);
        double z = StdIn.readDouble();

        transform(z);
        StdOut.println(z);
        StdOut.println(w + transform(x, y));
    }
}
```

The file `numbers.txt` contains the following three lines:

```
4
5
6
```

(a) What is printed when we run `Methodical` with the arguments and input below?

```
% java Methodical 1 2 3 < numbers.txt
```

First line: ____________________________________________

Second line: ___________________________________________

Third line: ____________________________________________

(b) What type of error occurs if we run this command?

```
% java Methodical 1 2 3 < numbers.txt | java Methodical
```

Circle one of I, II, III or IV.

I. No such element in `readString`

II. Array index out of bounds

III. Number format exception in `parseInt`

IV. Program runs forever
7. **Recursion** (6 points)

For the first four parts of this problem, you will investigate the behaviour of the recursive method defined by:

```java
public static void f(int n) {
    // print n
    System.out.print(n + " "); // space to separate the outputs

    // recursive calls, but when n is zero, acts as the base case
    for (int i = 0; i < n; i++) {
        f(i);
    }
}
```

(a) What is printed when you call `f(0)`?

Output: 

(b) What is printed when you call `f(1)`?

Output: 

(c) What is printed when you call `f(2)`?

Output: 

(d) What is printed when you call `f(3)`?

Output: 

(e) For this part, we ask instead about the method `g`:

```java
public static int g(int n) {
    if (n % 2 == 0) return n/10;
    return g(g(n/10));
}
```

What is the value of `g(3122013)`?

Value: ________________
8. **Recursive Graphics** (7 points)

Here is a method that draws squares recursively:

```java
public static void draw(int n, double x, double y, double r) {
    if (n==0) return; // base case
    draw(n-1, x, y, r/4);
    StdDraw.square(x, y, r); // draw a square
    draw(n-1, x - r/2, y, r/4);
    draw(n-1, x + r/2, y, r/4);
}
```

Below, we plot the picture produced when `draw(3, 0.5, 0.5, 0.5)` is called. It draws thirteen squares, which we have also **labelled** with dashed circles and arrows.

(a) What is the order in which the squares were drawn? Write all of the integers from 1 to 13 in the circles to indicate this order, with 1 labelling the first square drawn and 13 the last.

(b) Which of the follow expressions represents the order of growth of the running time of `draw` as a function of the first argument `n`? Circle one.

- \( \log_3 n \)
- \( n \log_3 n \)
- \( n^3 \)
- \( 3^n \)