This test has 10 questions worth a total of 50 points. You have 120 minutes. The exam is closed book, except that you are allowed to use a one page cheatsheet. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. 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.”

---------------------

Signature

Name:

NetID:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Sub 1</td>
<td></td>
<td>Sub 2</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

P01   TTh 1:30   Andrea
P01A  TTh 1:30   Sid
P01B  TTh 1:30   Woo Chang
P02   TTh 2:30   Forrest
P02A  TTh 2:30   Ananya
P03   TTh 3:30   Chang
P04   TTh 7:30   Tim
P05   WF 10     Yaping
P06   WF 11     Maia
P07   WF 1:30   Ganesh
P07A  WF 1:30   Sonya
P07B  WF 1:30   Yi
0. Miscellaneous. (1 point)

(a) Write your name and Princeton NetID in the space provided on the front of the exam, and circle your precept number.

(b) Write and sign the honor code on the front of the exam.

1. Number systems. (4 points)

(a) What is the decimal representation of the 16-bit two’s complement integer 1110012?
Circle your answer.

(b) Convert the decimal integer 200810 to hexadecimal. Circle your answer.

(c) Write the decimal integer -77 as an 8-bit two’s complement integer. Circle your answer.

(d) The absolute value in Java’s Math library computes the absolute value of an int (32-bit, two’s complement integer) as follows:

   public static int abs(int a) {
     if (a < 0) return -a;
     return a;
   }

   There is one value of x that makes Math.abs(x) return a negative integer. What is it? Circle your answer.

   \[ -2^{31} \quad -2^{31} - 1 \quad 0 \quad 2^{31} - 1 \quad 2^{31} \]
This page is intentionally left blank. Feel free to use for scratch work.
2. Nested loops and conditionals. (4 points)

Consider the following program.

```java
public class Triangle {
    public static void main(String[] args) {
        int N = Integer.parseInt(args[0]);
        for (int i = -N; i <= N; i++) {
            for (int j = -N; j <= N; j++) {
                if (i - j >= 0) System.out.print(i-j + " ");
                else System.out.print(" ");
            }
            System.out.println();
        }
    }
}
```

It takes a command-line parameter N and prints out a pattern, as below.

```
% java Triangle 3
0 . . . . . .
1 0 . . . . .
2 1 0 . . . .
3 2 1 0 . . .
4 3 2 1 0 . .
5 4 3 2 1 0 .
6 5 4 3 2 1 0
```

Describe in 10 words or less how to modify Triangle to print a mirror image, as below.

```
% java Triangle 3
 0 1 2 3 4 5 6
. 0 1 2 3 4 5
.. 0 1 2 3 4
... 0 1 2 3
.... 0 1 2
..... 0 1
....... 0
```
3. **Java loops and functions. (4 points)**

For each of the following code fragments, how many lines of output (number of calls to `System.out.println()`) does it produce? Circle your answers.

(a)

```java
for (int i = 0; i < 100; i++) {
    i = i*i*i;
    System.out.println("Princeton");
}
```

(b)

```java
public static void cube(int i) {
    i = i*i*i;
}

for (int i = 0; i < 100; i++) {
    cube(i);
    System.out.println("Tigers");
}
```
4. **Debugging and arrays. (8 points)**

Recall the *coupon collector problem*: repeatedly select one of N card types at random, and continue until you have collected one of each type. The program `Coupon.java` below attempts to simulate this process, printing out the total number of coupons collected.

```java
% more Coupon.java

public class CouponCollector {

    public static void main(void) {

        // read number of coupon types from command-line
        N = Integer.parseInt(args[0]);

        int cardcnt = 0; // number of cards collected
        int valcnt; // number of distinct cards collected

        // found[i] = true if card type i already collected
        boolean found = new boolean[N];

        // select cards at random, until you collect 1 of each type
        while (cardcnt > 0) {

            // select a new card type at random
            double val = (int) (Math.random() * N);
            cardcnt++;

            // check whether the card has already been collected
            if (found[i] = false)
                found[i] = true;
            valcnt++;

        }

        // print out total number of cards collected
        System.out.println(cardcnt);
    }
}
```

The code has at least 10 *independent* syntax and logical errors. Identify, circle, and correct 8 of them above.
5. Functions. (6 points)

(a) Given three positive integers $x$, $y$, $z$, you can form a triangle with the given side lengths if and only if each number is strictly less than the sum of the other two. Write a function `areTriangular()` that takes 3 integer arguments $x$, $y$, and $z$, and returns `false` if any number is greater than or equal to the sum of the other two, and `true` otherwise. Your answer will be graded for correctness and clarity.

```java
public static boolean areTriangular(int x, int y, int z) {
    // Function implementation
}
```

(b) List two compelling reasons why experienced programmers use functions.
6. **Standard input, standard output, and redirection.** (9 points)

Suppose that you have an input file containing a sequence of students and their grades. The input format consists of the number of assignments $N$, followed by a sequence of entries, each consisting of a *String* (the student’s last name) followed by $N$ integer scores.

```
% more data.txt
4
Ananya 96 78 61 83
Andrea 89 90 56 98
Chang 96 78 61 88
Forrest 87 78 61 93
Ganesh 96 78 61 83
Kevin 80 80 80 80
Maia 92 78 61 80
Sid 91 83 54 83
Sonya 94 78 72 81
Tim 88 78 64 83
Woo 87 85 66 83
Yaping 84 78 66 83
Yi 90 88 45 99
```

On the facing page, write a complete Java program `Assignments.java` that reads in data (in the given format) from standard input, and prints each student’s name and their average score to standard output. Assume that you have access to the library `StdIn.java`. Do *not* use arrays. Your answer will be graded for correctness and clarity.

Here is a sample execution.

```
% java Assignments < data.txt
Ananya 79.5
Andrea 83.25
Chang 80.75
Forrest 79.75
Ganesh 79.5
Kevin 80.0
Maia 77.75
Sid 77.75
Sonya 81.25
Tim 78.25
Woo 80.25
Yaping 77.75
Yi 80.5
```
(a)

```java
public class Assignments {
    public static void main(String[] args) {
        int N = StdIn.readInt(); // number of assignments
    }
}
```

(b) Suppose that you want to save the output of running program Assignments on the input file data.txt into an output file named output.txt. Write the command that you would use to do this.
7. **Recursive graphics. (6 points)**

Consider the following recursive Java function.

```java
public static void circles(int n, double x, double y, double size) {
    if (n == 0) return; // 1
    drawShadedCircle(x, y, size); // 2
    circles(n-1, x - size/3, y + size/2, size/2); // 3
    circles(n-1, x + size/3, y + size/2, size/2); // 4
}
```

The function call `circles(4, .5, .5, .5)` produces the output on the left.

(a) Give the order in which the 4 statements should appear in `circles()` so that the function call `circles(4, .5, .5, .5)` produces the output on the right. Circle your answer.

(b) Describe what would happen if the statements within `circles()` were ordered 4 3 2 1 and you called `circles(4, .5, .5, .5)`?
8. **TOY I. (2 points)**

(a) Approximately how many total bits of storage does the TOY machine have? Circle the best answer.

\[
2^8 \quad 2^{10} \quad 2^{12} \quad 2^{16} \quad 2^{256}
\]

(b) Which of the following are key features of the von Neumann architecture? Circle the best answer.

I. Data are encoded as sequences of bits.

II. Program and data are stored in the same shared memory.

III. Most widely used machine architecture today.

(a) I only.  
(b) I and II only.  
(c) I and III only.  
(d) I, II and III.  
(e) None.
9. **TOY II. (6 points)**

Consider the following TOY program, stored in memory locations 10 through 19.

10: 73FF  
    R[3] <- 00FF

11: 7101  
    R[1] <- 0001

12: 72AA  
    R[2] <- 00AA

13:  

14: A403  

15:  
    if (R[4] == 0) pc <- 18

16:  
    if (R[4] > 0) pc <- 13

17: 72BB  
    R[2] <- 00BB

18:  
    write R[2] to standard output

19: 0000  
    halt

(a) Fill in the missing machine code above for location 13, 15, 16 and 18.

(b) Run the program above, given the following values in memory locations FA through FE.

<table>
<thead>
<tr>
<th>Location</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FA</td>
<td>9117</td>
</tr>
<tr>
<td>FB</td>
<td>0000</td>
</tr>
<tr>
<td>FC</td>
<td>4004</td>
</tr>
<tr>
<td>FD</td>
<td>076F</td>
</tr>
<tr>
<td>FE</td>
<td>6FF0</td>
</tr>
</tbody>
</table>

What (if anything) is printed to standard output? Circle your answer.

(c) Give values for FA through FE that yield a different answer from (b).
TOY REFERENCE CARD

INSTRUCTION FORMATS

<table>
<thead>
<tr>
<th>. . . .</th>
<th>. . . .</th>
<th>. . . .</th>
<th>. . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format 1:</td>
<td>opcode</td>
<td>d</td>
<td>s</td>
</tr>
<tr>
<td>Format 2:</td>
<td>opcode</td>
<td>d</td>
<td>addr</td>
</tr>
</tbody>
</table>

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 \( R[d] \leftarrow R[s] \ll R[t] \)

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 \( \text{halt} \)
C: branch zero \( \text{if } (R[d] == 0) \ pc \leftarrow \text{addr} \)
D: branch positive \( \text{if } (R[d] > 0) \ pc \leftarrow \text{addr} \)
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 \( \text{mem}[FF] \) come from stdin.
Stores to \( \text{mem}[FF] \) go to stdout.

16-bit registers
16-bit memory locations
8-bit program counter