

CIS 120
2009 Placement Quiz

Name: _____

Student ID: _____
(from your PennCard)

Email _____ @seas.upenn.edu

Lab:(circle one)

201 202 203 204 205 206

	Score	Max
1		24
2		15
3		24
4		12
5		25
Total		100

The whole quiz is written using Java syntax, but only a tiny subset of the language is used, and we've tried to note any constructs that may be unfamiliar if your background is in some other language. In the questions that require writing code, don't worry about small details of concrete syntax (where semicolons and braces go, etc.) — just write it as you would in your favorite language.

If you are taking the quiz on your own, give yourself 50 minutes.

In multiple-choice questions, the “depends” choice is for cases in which the result depends on the values of the variables. Note that a result may seem to depend on the value of some variable, but in fact the result does not change with different values of the variable. In such case, the result *does not* depend on the value of the variable. Some other choices may not be a specific value, but instead “negative” or “positive” for a numeric result, “loops” if the code does not terminate, or “error” if the code fails without computing a result.

1. For each of the following *valid* expressions, circle what best describes the value computed by Java:

- (a) `2*3 - 3/2.0` 4 points
1 1.5 4.5 5

ANSWER: 4.5

- (b) `(7*3) * 3 == 7` 4 points
true false

ANSWER: false

- (c) `a < b && b < c && c < a` where a, b and c are `int` variables and `&&` is the boolean “and” operation 4 points
true false depends

ANSWER: false

- (d) `p || !p` where p is a boolean variable, `||` is the boolean “or” operation, and `!` is boolean “not” 4 points
true false depends

ANSWER: true

- (e) `x <= 0 || x >= 1 || x*x <= x` where x is a `double` (double-precision floating-point) variable 4 points
true false depends

ANSWER: true

- (f) `i < -100 || i > 100 || i * i > i` where i is an `int` variable 4 points
true false depends

ANSWER: depends

2. For each of the following program fragments write the final value of variable `x`. (The `=` operator is Java's notation for assignment to a variable.)

(a) `boolean y = true;`
 `boolean x = y;`
 `x = !x;`
 `y = y || !x;`

5 points

ANSWER: `false`

(b) `int y = 2;`
 `int x = y * 2;`
 `y = y * 2;`

5 points

ANSWER: `4`

(c) `String y = "hello";`
 `String x = y;`
 `y = "e";`

5 points

ANSWER: `"hello"`

3. For each of the following methods, circle the appropriate description of the return value. (The header “public int m(int x)” means “m is a procedure accepting an int argument x and returning an int.”)

(a)

```
public int m1() {
    int x = -1, y = 0, z = 0;
    if (x < 0) {
        y = y + 1;
        z = z + 1;
    } else {
        y = y + 1;
    }
    z = z + 1;
    return y+z;
}
```

1 2 3

6 points

ANSWER: 3

(b)

```
public int m2(int x) {
    boolean n = false;
    if (x > 0) { n = true; }
    if (n) { x = -x; }
    return x;
}
```

negative positive non-positive depends

6 points

ANSWER: non-positive

(c)

```
public int m3() {
    int f = 1;
    int i = 4;
    while (i > 0) {
        f = f * i;
    }
    return f;
}
```

1 24 loops

6 points

ANSWER: loops

(d)

```
public double m4 (int x) {
    int ans = 1;
    while (x > 1) {
        ans = ans * x;
        x = x - 1;
    }
    return ans;
}
```

1 error loops depends

6 points

ANSWER: depends

4. Suppose that we had defined a one-dimensional integer array `a` with four elements:

```
int[] a = {3, -1, -5, 4};
```

Circle the appropriate final value of `s` or other outcome of running the following code fragments. (Note that `for (int i = a; i < b; i = i + 1) {...}` is Java's notation for a loop with index variable `i` that begins from `a`, continues as long as `i < b`, and adds one to `i` at the end of each iteration.)

(a)

```
int s = 0;
for (int i = 0; i < 4; i = i + 1) {
    if (s >= 0) {
        s = s + a[i];
    }
}
```

6 points

error -3 0 1 2

ANSWER: -3

(b)

```
int s = 1;
for (int i = 1; i <= 4; i = i + 1) {
    s = s * (a[i] - a[i-1]);
}
```

6 points

error -16 -4 0 16

ANSWER: error

5. The following questions are about a `Matrix` class that encapsulates regular two-dimensional double-precision arrays. The class has the following outline:

```
public class Matrix {
    private int rows, cols;           \\ size of the matrix
    private double[][] data;         \\ contents of the matrix
    public Matrix(int m, int n) {     \\ construct a new matrix of size m*n
        rows = m;
        cols = n;
        data = new double[rows][cols];
    }
    public double get(int i, int j) { return data[i][j]; }
    public void set(int i, int j, double v) { data[i][j] = v; }
    public double[] sumRows() { ... }
    public void transpose() { ... }
}
```

The exact meaning of `public` and `private` here is not important, nor are the details of the object system — the point of this question is just to test your understanding of arrays and loops. The method `Matrix` constructs a new matrix object with the specified dimensions. The method `get` looks up the value in the `i`th row and `j`th column of the matrix by accessing the underlying array. The method `set` stores a value in a cell of the matrix.

In the following questions, we give incomplete methods with some white space that should be filled with the missing code. You should assume that by the time the methods are called, the two-dimensional array `data` exists and has dimension `rows` \times `cols`.

- (a) The method `sumRows` should return a one-dimensional array with elements containing the sum of the elements of the corresponding row in the matrix. **5 points**

```
public double[] sumRows() {
    double[] r = new double[rows];
    for (int i = 0; i < rows; i = i + 1) { // fill in the body of this loop

    }
    return r;
}
```

ANSWER:

```
for (int j=0; j<cols; j= j+1) {
    r[i] += get(i,j);
}
```

- (b) The method `getDiag` should return a one-dimensional array with elements containing the elements along the diagonal (i.e., the elements (0,0), (1,1), (2,2), etc.). (You should *not* assume that the matrix will be square.) 8 points

```
public double[] getDiag() {
```

```
}
```

ANSWER:

```
    int temp = cols;
    if (rows < cols) temp = rows;
    double[] d = new double[temp];
    for (int i = 0; i < temp; i++) {
        d[i] = data[i][i];
    }
    return d;
```

- (c) To *transpose* a *square* matrix (one with the same number of rows and columns), we swap the element at row i , column j with the element at row j , column i . Clearly, the *diagonal* elements, that is, those at row i and column i , stay put. For example, we show below a matrix, then the matrix with two pairs of elements swapped, and finally the transpose showing all pairs swapped. 12 points

before			
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

in between			
1	5	9	4
2	6	7	8
3	10	11	12
13	14	15	16

after			
1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

Complete the `transpose` method below that transposes the given matrix. If the matrix is not square, the method does nothing.

```
public void transpose() {
    if (rows == cols) {
        // fill in the initial value for i
        for (int i =          ; i < rows; i = i + 1) {

            // fill in the initial and loop termination test values for j
            for (int j =          ; j <          ; j = j + 1) {

                // perform the exchange of data[i][j] and data[j][i]

            }
        }
    }
}
```

ANSWER:

```
public void transpose() {
    if (rows == cols) {
        for (int i = 0 ; i < rows; i = i + 1) {
            for (int j = 0 ; j < i ; j = j + 1) {
                double temp = data[j][i];
                data[j][i] = data[i][j];
                data[i][j] = temp;
            }
        }
    }
}
```