# CIS 110 — Introduction To Computer Programming

**February 29, 2012 — Midterm**

**Answer key**

Scores:

| | |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| Total (100 max) | |

# CIS 110 Practice Midterm 1 Instructions

- **This practice midterm is substantially longer than the real midterm will be so that you see a wider range of practice questions. So don't panic at the length!**
- **All instructions after this one are the instructions that would appear on the actual test. If you are doing this practice test as a homework assignment, you may use books, notes, etc. *You can even work in groups!***
- You have 50 minutes to finish this exam. Time will begin when called by a proctor and end precisely 50 minutes after that time. If you continue writing after the time is called, you will receive a zero for the exam.
- This exam is *closed-book, closed-notes, and closed-computational devices.* Except where noted, you can assume that code included in the question is correct and use it as a reference for Java syntax.
- When writing code, the only abbreviations you may use are for `System.out.println` and `System.out.print` as follows:

  $$\text{System.out.println} \longrightarrow \text{S.O.PLN}$$
  $$\text{System.out.print} \longrightarrow \text{S.O.P}$$

  Otherwise all code must be written out as normal.
- Please do not separate the pages of the exam. If a page becomes loose, please make sure to write your name on it so that we don't lose it, and use the provided staplers to reattach the sheet when you turn in your exam.
- If you require extra paper, please use the backs of the exam pages or the extra sheet paper provided at the end of the exam. Clearly indicate on the question page where the graders can find the remainder of your work (e.g. "back of page" or "on extra sheet").
- If you have any questions, please raise your hand and an exam proctor will come to answer them.
- When you turn in your exam, you will be required to show ID. If you forgot to bring your ID, please talk to an exam proctor immediately.

*Good luck, have fun!*

## Miscellaneous

1. (1 points)

(a) Write your name, recitation number, and PennKey (username) on the front of the exam.

(b) Sign the certification that you comply with the Penn Academic Integrity Code

## Types and Casts

2. (7 points) Give the type and value of each of the following Java expressions. If an expression will not compile, write `Illegal` under type and put an X in value. You must fill in every entry. Entries left blank will be marked incorrect.

| Expression | Type | Value |
|---|---|---|
| `(7 / 2) * 2.0` | `double` | `6.0` |
| `(7 / 2.0) * 2` | `double` | `7.0` |
| `!(!!false && !!true)` | `boolean` | `true` |
| `"1.1" + "2.2"` | `string` | `"1.12.2"` |
| `1 < 2 < 3` | `Illegal` | X |
| `1 + 2.2 + "1.1" + "a"` | `String` | `"3.21.1a"` |
| `Integer.parseInt("123")` | `int` | `123` |

## Java Expressions

3. (3 points) Assume that `a`, `b`, and `c` are variables of type `boolean`. Consider the following three conditions:

(a) `(a || b) && c` X

(b) `(a && b) || c` X

(c) `!(a && b) && c` X

Which of the expression(s) above is (are) always true if `a` and `b` have different values (i.e. one is true and one is false) and `c` is true? Circle the expressions that evaluate to true.

## Array Declarations

4. (8 points) Among the following code fragments, circle the ones that will *not* cause a compile-time error.

(a) `int[] a = int[10];`

(b) `int[] a = new int[10];` X

(c) `int[10] a;`

(d) `int[10] a = new int[10];`

(e) `int[] a = {1, 2, 3}; int b = a;`

(f) `int[] a;` X

(g) `int a = {1, 2, 3};`

(h) `int[] a = {1, 2, 3}; int[] b = a;` X

## Syntax and Scope

5. (7 points) Consider the following code:

```
public class Cubes
{
  public static int square(int i)
  {  return i * i * i; }

  public static void main(String[] args)
  {
    for (int i = 1; i <= 1000; i++)
      StdOut.println(square(i));
  }
}
```

Among the following statements, circle those that are true:

(a) Will not compile beacuse braces in `square()` are not on separate lines.

(b) Will not compile because braces are missing in the `for` loop.

(c) Will not compile because `i` is not declared in `square()`.

(d) Prints only a few lines because `square()` rapidly increases the `i` used by main.

(e) Prints the squares of the integers from 1 to 1000.

(f) Prints the cubes of the integers from 1 to 1000. X

(g) Goes into an infinite loop.

## Debugging

6. (9 points) Consider the following code snippet, which is intended to examine an array of of `double` values and print out the minimum value in the array and its location. The line numbers at the left are for your convenience in referring to the code. You may assume `arr[]` is an array of `double` values that has been properly declared and initialized.

```
 1    int N = arr.length;
 2    double min = 0;
 3    int minLocation=0;
 4
 5    for (int i = 1; i <= N; i++) {
 6       if (arr[i] < min)
 7          min = arr[i];
 8          minLocation = i;
 9    }
10
11    System.out.print("The minimal value is arr[");
12    System.out.println(minLocation + "] = " + min);
```

In the spaces provide below, identify (*in ten words or less*, each) three bugs in the code that will result in either an incorrect answer being printed or a program crash. Circle your answers.

(a) A bug:
`for` loop should go from `0` to `N-1` on line 5

(b) Another bug:
`double min = Double.POSITIVE_INFINITY` on line 2
`int minLocation = -1` on line 3

(c) A third bug:
`if` statement needs curly braces around lines 7 and 8

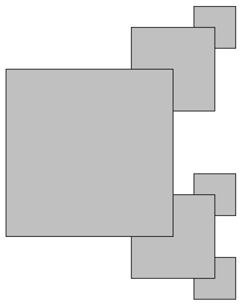
## Recursive Graphics

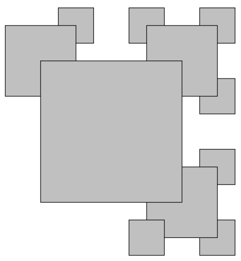
7. (6 points) Consider the following recursive method:

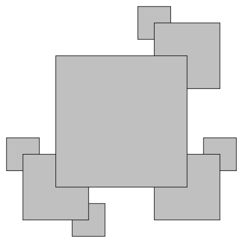
```
public static void squareLife(double x, double y, double r) {
  if (r < 0.05) return;

  // Select a random integer from the set [0, 1, 2, 3, 4]
  int randomInt = StdRandom.uniform(5);

  if (randomInt < 1)
    squareLife(x - r, y - r, r / 2); // bottom left
  if (randomInt < 2)
    squareLife(x - r, y + r, r / 2); // top left
  if (randomInt < 3)
    squareLife(x + r, y - r, r / 2); // bottom right
  if (randomInt < 4)
    squareLife(x + r, y + r, r / 2); // top right

  drawShadedSquare(x, y, r);
}
```

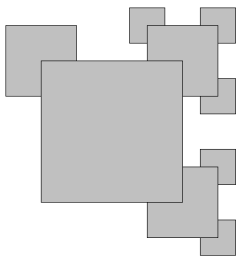
Assume that the helper method `drawShadedSquare()` draws a gray shaded square of radius `r` that is outlined in black and centered on `(x, y)`.

(a) Suppose that a client program issues the call `squareLife(0.5, 0.5, 0.25)`. Circle the figures that could possibly result.


this one






this one

(b) Suppose that a client program issues the call `squareLife(0.5, 0.5, 0.6)`. What is the maximum possible number of times `drawShadedSquare()` could be called as a result? Write your answer in the blank provided.

Maximum number of squares drawn: 85

## Recursive Methods

8. (3 points) The function below is supposed to sum all the odd, or all the even, positive integers up to n, depending on whether n is odd or even.

```
public static int foo(int n) {
    if (n == 0 || n == 1) return n;
    return n + foo(n - 2);
}
```

Circle any of the following that are correct:

(a) base case written incorrectly so function will get the wrong answer, or no answer, for some valid positive values of n.

(b) reduction step written incorrectly, so function will get the wrong answer, or no answer, for some valid, positive values of n

(c) function is tail-recursive and therefore will get the wrong answer, or no answer, for some valid positive values of n

(d) this function looks fine to me X

(e) it doesn't have problems a, b, or c, but does have the following problem, which will cause it to get the wrong answer, or no answer, for some valid positive values of n (write the problem down in at most one sentence)

## Array Whisperer

9. (20 points) Write a function `duplicate` that takes an integer array `arr` and an integer n as arguments and returns a new integer array that contains `n` copies of `arr` laid out end-to-end. Assume that you are passed a valid array whose length is at least 0 and that `n` is positive. Here are some example invocations of `duplicate` and their results:

| Array | n | Return Value |
|---|---|---|
| { 1, 2, 3 } | 3 | { 1, 2, 3, 1, 2, 3, 1, 2, 3 } |
| { 0 } | 5 | { 0, 0, 0, 0, 0 } |
| { 1, 2, 3, 4, 5 } | 1 | { 1, 2, 3, 4, 5 } |

```
public static int[] duplicate(int[] arr, int n) {
    int[] out = new int[arr.length * n];

    for (int i = 0; i < n; i++)
        for (int j = 0; j < arr.length; j++)
            out[i * arr.length + j] = arr[j];

    return out;
}
```

## Prime Numbers

10. (20 points) Write a function `primes` that takes an integer `N` and prints out the number of prime numbers less than `N`. Recall that a prime number is an integer with exactly two divisors: one and itself. (The smallest prime number is therefore 2.) Hint: use nested `for` loops and the `%` operator.

```
public static void primes(int N) {
    int count = 0;
    for (int i = 2; i < N; i++) {
        boolean prime = true;
        for (int j = 2; j * j <= i; j++) {
            if (i % j == 0)
                prime = false;
        }
        if (prime) count++;
    }

    System.out.println(count);
}
```

## Be Assertive

11. (15 points) For each of the labeled points in the code fragment below, identify each of the assertions in the table as being *always* true, *never* true, or *sometimes* true and sometimes false. Assume that the call to `StdIn.readInt()` always succeeds and that the values of all `int`s stay within the valid range for integers (i.e. no value will grow so large that it will wrap around and become negative, or vice versa).

*Note:* You may abbreviate always with A, never with N, and sometimes with S.

```
public static int foo(int x) {
    int y = x;
    int z = 0;
    boolean b = true;
    // POINT A
    while (b) {
        // POINT B
        y = StdIn.readInt();
        if (x < y) {
            z = z + y - x;
            // POINT C
        } else if (x == y) {
            z = z - y + x;
            b = false;
            // POINT D
        }
    }
    // POINT E
    return z;
}
```

| | `b == true` | `x == y` | `z > 0` |
|---|---|---|---|
| A | A | A | N |
| B | A | S | S |
| C | A | N | A |
| D | N | A | S |
| E | N | A | S |

## Postscript (extra paper)