CIS 110 — Introduction to Computer Programming

February 29, 2012 — Midterm

Answer key

Scores:

1	
2	
3	
4	
5	
6	
Total (50 max)	

CIS 110 Midterm Instructions

- 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.
- This exam is long. If you get stuck part way through a problem, it may be to your advantage to go on to another problem and come back later if you have time.
- 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(s) of 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. (5 points) Give the type and value of each of the following Java expressions. If an expression will crash or 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
`2 / (1 / 2)`	`Illegal`	`X`
`2.0 / (1 / 2)`	`double`	`Infinity`
`2.0 / (1.0 / 2)`	`double`	`4.0`
`2.0 / (1.0 / (int) 2)`	`double`	`4.0`
`2.0 / (1.0 / 2) + ""`	`String`	`"4.0"`

Debugging

3. (8 points) Consider the following function `awol` which returns a fixed permutation (rearrangement) of the character array `foo` if `foo` is the same length as the `permute` array. Otherwise it returns an empty `char` array. You may assume `foo` is a valid array of `char`s.

Note: The line numbers are not part of the program. They are included for your convenience.

```
 1 public static char[] awol(char[] foo) {
 2     int permute = { 8, 4, 2, 3, 6, 10, 1, 9, 5, 0, 11, 7 };
 3     char[] bar = char[permute.length];
 4
 5     if (foo.length == permute.length) return;
 6     else for (int i = 0; i < foo.length; i++) {
 7         bar[i] = foo[permute[i]];
 8     }
 9
10     return bar;
11 }
```

Three lines of this program contain errors that will cause the program to return an incorrect result, crash on certain inputs, or not compile. Provide the corrected versions of the buggy code below. You only need to rewrite enough of the line to correct the bug. Circle your answer.

(a) A corrected line of code:
`int[] permute = ...` on line 2

(b) Another corrected line of code:
`char[] bar = new char[permute.length]` on line 3

(c) A third line of code:
`if (foo.length != permute.length) return new char[0];` on line 5

(d) What will the corrected version of `awol` return when it is passed the array
{`'p', 'r', 'i', 'v', 'a', 't', 'e', 's', 'n', 'a', 'f', 'u'` } as an argument? For brevity, you may write your answer as a string rather than an array.
`naivefratpus`

Recursive Graphics

4. (7 points) Consider the following recursive method:

```
public static void bongo(double x, double y, double r, boolean odd) {
    drawShadedSquare(x, y, r);

    if (r < 0.01) return;

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

    double new_r = 0.5 * r;
    if (odd) new_r = 0.4 * r;

    if (randomInt < 1)
        bongo(x - r, y - r, new_r, !odd); // bottom left
    if (randomInt < 2)
        bongo(x - r, y + r, new_r, !odd); // top left
    else if (randomInt < 3)
        bongo(x + r, y - r, new_r, !odd); // bottom right
    if (randomInt < 4)
        bongo(x + r, y + r, new_r, !odd); // top right

  return;
}
```

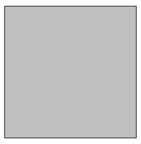
Assume that the helper method `drawShadedSquare()` draws a gray shaded square of radius `r` that is outlined in black and centered on `(x, y)`.

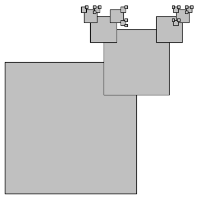
If a client issues the call `bongo(0.5, 0.5, 0.25, false)`:

(a) What is the radius of the smallest square that could possibly be drawn. *Hint:* Compute the radius using fractions ($\frac{1}{2}$) rather than decimals (0.5). Write your answer in the space provided.

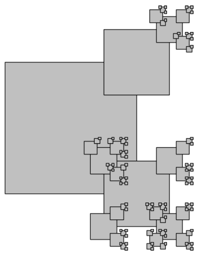
Radius of the smallest radius: 0.005

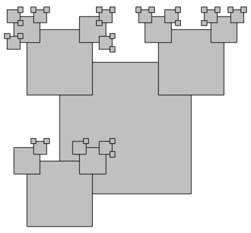
(b) Which of the figures below could possibly be drawn? Circle your answers.

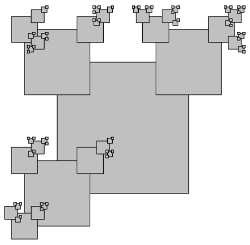




this one







this one

Tracery

5. (9 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 `bar` is only called from within `foo`, 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).

Abbreviate always with **A**, never with **N**, and sometimes with **S**.

```
public static int foo(int x) {
  if (x == 0) return 0;

  int y = 3 * x;

  // POINT A

  if (y > x) {
    // POINT B
    return bar(y);
  } else {
    // POINT C
    return bar(x * y);
  }
}

public static int bar(int x) {
  int y = x / 2;

  // POINT D
  if (2 * y == x) {
    // POINT E
    return foo(y / 4);
  }

  // POINT F
  return x + foo(y / x);
}
```

	x > 0	y > 0	y is even
A	S	S	S
B	A	A	S
C	N	N	S
D	A	A	S
E	A	A	S
F	A	A	S

Vegas, Here We Come

6. (20 points)

Write a function `shuffle` that takes an integer array `deck` and returns a new integer array that is a shuffled version of `deck`, in which elements from the second half of `deck` are perfectly interleaved with the elements from the first half. If `deck` contains an odd number of elements, the central element should end up as the last element of the shuffled array. You can assume that `shuffle` receives a valid integer array, but it might be of length 0. Here are some example invocations of `shuffle` and their results:

Array	Return Value
{ }	{ }
{ 1, 2, 3, 4, 5 }	{ 1, 4, 2, 5, 3 }
{ 1, 2, 3, 4, 5, 6, 7, 8 }	{ 1, 5, 2, 6, 3, 7, 4, 8 }

```
public static int[] shuffle(int[] deck) {
    int[] out = new int[deck.length];

    int mid = (deck.length + 1) / 2;
    for (int i = 0; i < out.length; i = i + 2)
        out[i] = deck[i / 2];
    for (int i = 1; i < out.length; i = i + 2)
        out[i] = deck[mid + i / 2];

    return out;
}
```

Postscript (extra paper)