CIS 110 — Introduction to Computer Programming

7 June 2012 — Midterm

Name: ______________________________

Recitation # (e.g. 201): ______________________________

Pennkey (e.g. bjbrown): ______________________________

My signature below certifies that I have complied with the University of Pennsylvania's Code of Academic Integrity in completing this examination.

______________________________ ____________________

Signature Date

Scores:

1		1
2		5
3		6
4		13
5		15
6		20
Total:		60

CIS 110 Final Instructions

- You have 90 minutes to finish this exam. Time will begin when called by a proctor and end precisely 90 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`, `System.out.print`, and `System.out.printf` as follows:

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

  Otherwise all code must be written out as normal, including all curly braces and semicolons.
- Please do not separate the pages of the exam. If a page becomes loose, write your name on it and use the provided staplers to reattach the sheet when you turn in your exam so that we don't lose it.
- 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"). Staple an extra sheets you use to the back of your exam when you turn it in using the provided staplers.
- 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 may 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

Find the Bugs

2. (5 points) Identify five errors in the following sorting routine that prevent it from compiling. The line numbers are included for your convenience and are not part of the program.

```
 1:  public static int selectionSort(int[] input) {
 2:    for (int i = 0; i < input.length; i++)
 3:      int minIndex = i;
 4:      for (int j = i; j < input.length; j++) {
 5:        if (input[j] < input[minIndex])
 6:          int minIndex = j;
 7:      }
 8:
 9:      int temp = input[i];
10:      input[i] = input[minIndex]
11:      input[minIndex] = temp;
12:    }
13:
14:    return input[];
15:  }
```

Bug 1:

Bug 2:

Bug 3:

Bug 4:

Bug 5:

Types and Expressions

3. (6 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.

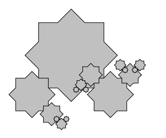
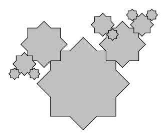
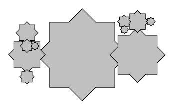
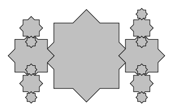
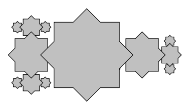
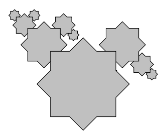
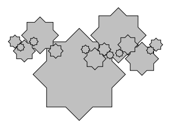
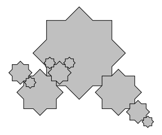
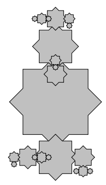
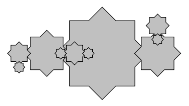
Expression	Type	Value
`3 < 4 < 5`		
`5 / 2`		
`"5" + Math.max(3, 4)`		
`(double) 2 / 4`		
`(!!true) && (!!(!false))`		
`Integer.parseInt(2)`		

Recursive Graphics

4. (13 points) Consider the following recursive function, then answer the questions on the following page. Assume that the helper function `drawShadedStar()` draws an eight-pointed, shaded gray star of radius `r` that is outlined in black and centered on `(x, y)`.

```
public static void starryNight(double x, double y, double r, boolean odd) {
  // Select a random integer from the set [0, 1, 2, 3]
  int randomInt = (int) (4 * Math.random());

  double new_r  = 0.5 * r;
  double offset = 0.6 * r;

  if (r < 0.02) return;

  drawShadedStar(x, y, r);

  if (randomInt % 2 == 0)
    starryNight(x - offset, y - offset, new_r, !odd);
  else if (randomInt / 2 == 0)
    starryNight(x + offset, y + offset, new_r, !odd);
  if (randomInt / 2 == 1)
    starryNight(x + offset, y - offset, new_r, !odd);
  else if (randomInt % 2 == 1)
    starryNight(x - offset, y + offset, new_r, !odd);
}
```

Assume a client issues the call `starryNight(0.5, 0.5, 0.25, true)`. For each figure below, state that it is **correct**, or indicate what feature(s) of the figure make it incorrect. Each inccorect figure has at most two problems.

Tracery

5. (15 points)

For each of the labeled points in the code fragment below, identify each of the assertions in the table as being *sometimes*, *always*, or *never* true. 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 become negative, or vice cersa).

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

```
public static int foo(int x, int y, int z) {
  x = x * x;
  if (x < 0) y = y * y;
  else y = y - 2 * y;
  // POINT A

  if (x < 0) z = y;
  else y++;
  // POINT B

  if (z < 0) z = x;

  if (x > y * y) {
    // POINT C
    return bar(y / 3, x + 1, z - 1);
  } else {
    // POINT D
    return bar(y + 1, x, z - 1);
  }
}

public static int bar(int x, int y, int z) {
  if (z == 0) return 42;

  x = x * 4;
  // POINT E
  return foo(x / 2, -y - Math.abs(x), z - 1);
}
```

	`x >= 0`	`y > x`	`z < 0`
A			
B			
C			
D			
E			

Partial Sums

6. (20 points) Write a function `sumUpTo` that takes a single argument `N`, reads in `N` integers from standard input using `StdIn.readInt()`, and returns an integer array of length `N` where the value at index `i` is the sum of the last `i + 1` numbers read. For example, if you read in the values 1, 3, -1, 4, 2, you should produce and return an array with the values 2, $2+4$, $2+4+(-1)$, $2+4+(-1)+3$, $2+4+(-1)+3+1$, i.e. `{2, 6, 5, 8, 9}` . Any implementation that works will receive a good grade, but for full credit your function should create only one array. You may assume that `N` is at least 1 and that `StdIn.readInt()` always succeeds.

Postscript (extra paper)

Postscript (extra paper)