CSE 143X: Accelerated Computer Programming I/II Autumn 2015

Midterm Exam Solutions

Name:	**Sample Solutions**		
ID #:	**1234567**		
TA:	**The Best**	**Section:**	**A9**

INSTRUCTIONS:

- You have **50 minutes** to complete the exam.
- You *will* receive a deduction if you keep working after the instructor calls for papers.
- This exam is closed-book and closed-notes. You may not use any computing devices including calculators.
- Code will be graded on proper behavior/output and not on style, unless otherwise indicated.
- Do not abbreviate code, such as "ditto" marks or dot-dot-dot ("...") marks. The only abbreviations that are allowed for this exam are: `S.o.p` for `System.out.print` and `S.o.pln` for `System.out.println`.
- You do not need to write import statements in your code.
- You may not use extra scratch paper on this exam. Use the provided spaces for extra work.
- If you write work you want graded on a strange page, clearly label it.
- If you enter the room, you must turn in an exam before leaving the room.
- You must show your Student ID to a TA or instructor for your exam to be accepted.
- If you get stuck on a problem, move on and come back to it later.

Problem	Points	Score	Problem	Points	Score
1	12		4	15	
2	15		5	18	
3	12		6	18	
			Σ	90	

Mechanical Missions.

This section tests whether you are able to trace through code of various types in the same way a computer would.

1. Mystery(Parameter, Parameter) [12 points]

```
1  public class Mystery {
2     public static void main(String[] args) {
3        String they = "like";
4        String totally = "can";
5        String yes = "they";
6        String like = "yes";
7
8        mumble(like, yes, totally);
9        mumble(totally, "Shannon", "maybe");
10       mumble(totally, they, yes);
11       mumble("can't not", "Michael", "no");
12    }
13
14    public static void mumble(String can, String they, String yes) {
15       System.out.println(they + " totally " + yes + " like " + can);
16    }
17 }
```

For each line of output, write down what is printed out.

	Line	Text Printed
(a)	Line 1	they totally can like yes
(b)	Line 2	Shannon totally maybe like can
(c)	Line 3	like totally they like can
(d)	Line 4	Michael totally no like can't not

2. assert(true) [15 points]

In this question, you will identify various assertions as being either always true, never true or sometimes true/sometimes false at various points in program execution. The comments in the method below indicate the points of interest.

```
1  public static void mystery(int x, int y) {
2     int z = 0;
3     // Point A
4
5     while (x < y) {
6        // Point B
7
8        z++;
9        if (z % 2 == 0) {
10          x = x * 2;
11          // Point C
12       }
13       else {
14          y--;
15          // Point D
16       }
17    }
18    // Point E
19    System.out.println(z);
20 }
```

Fill in the table below with the words ALWAYS/SOMETIMES/NEVER. You may abbreviate them.

	`x < y`	`z == 0`	`z % 2 == 0`
Point A	**S**	**A**	**A**
Point B	**A**	**S**	**S**
Point C	**S**	**N**	**A**
Point D	**S**	**N**	**N**
Point E	**N**	**S**	**S**

3. Semantics[a] [12 points]

```
1 public class Point {
2    int x;
3    int y;
4 }
```

```
1 public class PointBox {
2    Point p;
3 }
```

```
1  public class ReferenceMystery {
2     public static void main(String[] args) {
3        int a = 3;
4
5        Point p = new Point();
6        p.x = 10;
7        p.y = 5;
8
9        PointBox b1 = new PointBox();
10       PointBox b2 = new PointBox();
11       b1.p = p;
12       b2.p = p;
13
14       a = rotate(a, b1);
15       System.out.println(a + " " + b1.p.x + " " + b1.p.y + " " + b2.p.x + " " + b2.p.y);
16       rotate(a, b2);
17       System.out.println(a + " " + b1.p.x + " " + b1.p.y + " " + b2.p.x + " " + b2.p.y);
18    }
19
20    public static int rotate(int a, PointBox b1) {
21       int tmp = a;
22       a = b1.p.x;
23       b1.p.x = b1.p.y;
24       b1.p.y = tmp;
25       System.out.println(a + " " + b1.p.x + " " + b1.p.y);
26       return a;
27    }
```

For each line of output, write down what is printed out.

	Line	Text Printed
(a)	**Line 1**	**10 5 3**
(b)	**Line 2**	**10 5 3 5 3**
(c)	**Line 3**	**5 3 10**
(d)	**Line 4**	**10 3 10 3 10**

Programming Pursuits.

This section tests whether you synthesized various topics well enough to write novel programs using those topics.

4. Boxed In [15 points]

Write a method called `printBox` that takes a `String` (`side`) as a parameter and prints a square with dimensions equal to the length of `side`. Your method should use `side` as the text for each of the four sides of the box.

Example Output

```
printBox("***")   ------->   ***
                              * *
                              ***

printBox("abcde")  ------->   abcde
                              b   d
                              c   c
                              d   b
                              edcba
```

Implementation Restrictions

- You may not define auxiliary structures (no extra Strings or arrays).
- You may (but are not required to) define extra static methods to reduce redundancy.
- You may assume that the input `String` is at least one character long.
- You may only use the `String` methods on the cheatsheet.

Solution:

```
1  public static void printBox(String side) {
2     if (side.length() > 1) {
3        System.out.println(side);
4     }
5
6     for (int i = 1; i < side.length() - 1; i++) {
7        System.out.print(side.charAt(i));
8        for (int j = 0; j < side.length() - 2; j++) {
9           System.out.print(" ");
10       }
11       System.out.println(side.charAt(side.length() - 1 - i));
12    }
13
14
15    for (int i = side.length() - 1; i >= 0; i--) {
16       System.out.print(side.charAt(i));
17    }
18    System.out.println();
19 }
```

5. 1 + 1 = 2 [18 points]

An arithmetic sequence is an array of numbers where the difference between every adjacent pair of integers is the same constant. Write a method called `isArithmeticSequence` that takes an `int` array as a parameter and returns `true` if the array is a valid arithmetic sequence and `false` otherwise. Arrays with fewer than two elements should be considered arithmetic sequences.

Example Output

Method Call	Return Value
`isArithmeticSequence({1, 2, 3, 4})`	`true`
`isArithmeticSequence({28, 32, 36, 40})`	`true`
`isArithmeticSequence({20, 10, 0, -10})`	`true`
`isArithmeticSequence({2, 3, 5, 7, 11})`	`false`

Solution:

```
1  public static boolean isArithmeticSequence(int[] arr) {
2     if (arr.length < 2) {
3        return true;
4     }
5     int diff = arr[1] - arr[0];
6     for (int i = 0; i < arr.length - 1; i++) {
7        if (arr[i + 1] - arr[i] != diff) {
8           return false;
9        }
10    }
11    return true;
12 }
```

6. Single File Please! [18 points]

Write a method called `gradeMCTest` that takes two `Scanners` as parameters: `students` and `correct` which represent student answers to a multiple choice test and the correct answers to the same test, respectively.

For example, the file `students` points to might look like:

```
>> ABCAC
>> ACCDC
>> CCCCC
```

Notice this input has five questions and three students to grade.

And the file `correct` points to might look like:

```
>> A
>> C
>> B
>> D
>> A
```

Notice each answer is on its own line, and the entire file represents a single set of answers to the test.

To "grade" a multiple choice test, for each student, compare the correct answer for each question with their answer, and count the number of correct responses. Your method should output the number of correct answers for each student, one student per line.

The correct output to the inputs above would be:

```
>> 1
>> 3
>> 1
```

Implementation Restrictions

- You may (but are not required to) define extra static methods to reduce redundancy.
- You may assume that the input `Scanners` point to valid inputs (e.g., the right length for each line).
- You may not assume a particular number of questions or a particular number of students.

Solution:

```
1  public static void gradeMCTest(Scanner students, Scanner correct) {
2     String answers = "";
3     while (correct.hasNext()) {
4        answers += correct.next();
5     }
6
7     while (students.hasNextLine()) {
8        String student = students.nextLine();
9        int result = 0;
10       for (int i = 0; i < answers.length(); i++) {
11          if (student.charAt(i) == answers.charAt(i)) {
12             result++;
13          }
14       }
15       System.out.println(result);
16    }
17 }
```