CIS 110 — Introduction to Computer Programming
Spring 2016 — Midterm

Name: _____

Recitation # (e.g., 201): _____

Pennkey (e.g., eeaton): _____

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

_____ _____

Signature **Date**

Instructions:

- **Do not open this exam until told by the proctor**. You will have exactly 110 minutes to finish it.
- **Make sure your phone is turned OFF (not to vibrate!) before the exam starts.**
- Food, gum, and drink are strictly forbidden.
- **You may not use your phone or open your bag for any reason**, including to retrieve or put away pens or pencils, until you have left the exam room.
- This exam is *closed-book, closed-notes, and closed computational devices*.
- If you get stuck on a problem, it may be to your benefit to move on to another question and come back later.
- All answers must be written on the exam booklet; answers written on scrap paper will not be counted.
- All code must be written out in proper java format, including all curly braces and semicolons.
- Do not separate the pages. If a page becomes loose, re-attach it with the provided staplers.
- Staple all scratch paper to your exam. Do not take any sheets of paper with you.
- If you require extra paper, please use the backs of the exam pages or the extra pages 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").**
- Use a pencil, or blue or black pen to complete the exam.
- If you have any questions, raise your hand and a 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, talk to an exam proctor immediately.**
- We wish you the best of luck.

Scores:

[For instructor use only]

Cover Page		1 pt
Section 1		8 pts
Section 2		11 pts
Section 3		14 pts
Question 4.1		8 pts
Question 4.2		10 pts
Section 5		10 pts
Section 6		10 pts
Section 7		8 pts
Total:		**80 pts**

0. (1 pt) Cover Page Information:

- Check that your exam has all 9 pages (excluding the cover page and scrap paper).
- Write your name, recitation number, and PennKey (username) on the front of the exam.
- Sign the certification that you comply with the Penn Academic Integrity Code.

SECTION 1: MISCELLANEOUS (8 pts total)

Circle the output of the code snippets. In case of any error in the code, choose option E.

1.1. (2 pts)
```
int a = (int) 'A';
System.out.println((char) a);
```

A. 65 B. A C. a D. None of the above E. Error

1.2. (2 pts)
```
int b = (int) 4.5;
double c = 2.0;
System.out.println(b / c);
```

A. 2 B. 2.25 C. 2.0 D. None of the above E. Error

1.3. (2 pts)
```
int a = 3 * 5 % 7 + 2;
System.out.println(a);
```

A. 6 B. 17 C. 21 D. 3 E. Error

1.4. (2 pts)
```
String[] arr = {"hello", "goodbye", "see ya"};
arr[2] = null;
System.out.println(arr.length);
```

A. 1 B. 3 C. 2 D. None of the above E. Error

SECTION 2: LOOPS AND CONDITIONALS (11 pts total)

2.1. Fill in the blanks in the code below to produce the output: "3, 9, 27, 81, "

```
public static void fillInTheFunction(){
    int n = 1;
    while(   A   ){
        n =   B   ;
        System.out.print(n + ", ");
    }
    System.out.println();
}
```

Blank A: _____ **(2 pts)**

Blank B: _____ **(1 pts)**

2.2. **(8 pts; 2 pts each)** Match the pattern with the function that most likely produced it. If none of the functions match the pattern, write "**None**" in its corresponding blank; otherwise, write the letter corresponding to the matching function.

i.
```
12345
2345
345
45
5
```

ii.
```
1
12
123
1234
12345
```

iii.
```
1
123
12345
123
1
```

iv.
```
1
123
12345
```

i. _____

ii. _____

iii. _____

iv. _____

A.
```
public static void print() {
     int a = 5, num = 1, b = 5;
     for(int i = 1; i <= a; i++) {
          for(int j = a - i; j >= 1; j--){
               System.out.print(" ");
          }
          for(int k = 1; k <= b; k++){
               System.out.print(num);
               num++;
          }
          System.out.println();
          num = 1;
     }
}
```

B.
```
public static void print() {
     int a = 5, num = 1, b = 5;
     for(int i = 1; i <= a; i++) {
          System.out.println();
          for(int j = 1; j <= b; j++){
               System.out.print(num);
               num++;
          }
          num = i + 1;
          b--;
     }
}
```

C.
```
public static void print() {
     int a = 5, num = 1, b = 4;
     for(int i = 1; i <= a; i++){
          for(int j = 1; j <= b; j++){
               System.out.print(" ");
          }
          b--;
          for(int k = 1; k <= i; k++){
               System.out.print(num);
               num++;
          }
          System.out.println();
          num = 1;
     }
}
```

D.
```
public static void print() {
     int a = 3, num = 1;
     for(int i = 1; i <= a; i++){
        for(int j = a - i; j >= 1; j--){
             System.out.print(" ");
        }
        int b = i * 2 - 1;
        for(int k = 1; k <= b; k++){
             System.out.print(num);
             num++;
        }
        System.out.println();
        num = 1;
     }
}
```

SECTION 3: VARIABLES, OPERATORS, & TYPES (14 pts total)

3.1. (14 pts) For each code fragment **1.)** Fill in the most appropriate data type for ??? in the 1st column and **2.)** give the value that *z* contains after the code has been executed in the 2nd column. If the code would result in an error, write "ERROR" in the 1st column and give the reason for the error in the 2nd column.

Code	**DataType/Error**	**Value of z/reason**
??? z = 9; z++;	int	z = 10
int x = 100; ??? z = x.length;	ERROR	x doesn't have a x.length field
int x = 5; ??? z = x / 2;		
int x = 0; **int** y = 1; ??? z = (x = y);		
String x = "CIS"; ??? z = x + 1 + 1 + 0;		
??? z = Double.parseDouble(2.0);		
int[] x = {2, 5, 1, 0}; ??? z = x[x[0]];		
String x = "Gorilla"; ??? z = x.charAt(6);		
??? z = 'a' < 'b';		

SECTION 4: FUNCTIONS! (18 pts total)

4.1. (8 pts) Write the non-recursive static function `mutantBunnyEars` based on the function header below. For full credit, write the code as efficiently as possible:

```
/** Function Name: mutantBunnyEars
 *  Parameters:
 *    n - number of mutant bunnies in a line
 *  Returns:
 *    the number of ears in the mutant bunny line
 *    assuming that even numbered mutant bunnies have 2 ears
 *    and odd numbered mutant bunnies have 3 ears.
 *  Examples: mutantBunnyEars(1) would return 3
 *            mutantBunnyEars(2) would return 3+2 = 5
 *            mutantBunnyEars(3) would return 3+2+3 = 8
 */
```

4.2. (10 pts) Write the static function `numIncreases` based on the header below. Your implementation may be recursive or non-recursive.

```
/** Function Name: numIncreases
 *  Function Description:
 *    Returns the number of times consecutive elements of the
 *    given array increase, reading left to right.
 *
 *    For example, the array arr = {1,0,2,3,3} contains two (2)
 *    increases (moving left to right though the array):
 *      1.) arr[1] to arr[2], since 0 < 2
 *      2.) arr[2] to arr[3], since 2 < 3
 *    All other consecutive elements decrease or remain the same.
 *    So, numIncreases({1,0,2,3,3}) returns 2
 *  Parameters:
 *    arr - array of integers
 *  Returns:
 *    the total number of increases
 *  Error Checking:
 *    if the array is null, return 0
 */
```

SECTION 5: DEBUGGING (10 pts total)

The following program deals 5-card hands at random to the given number of players.

```
java Deal 3
```

However, it contains a number of bugs. Find and correct the bugs by filling in the table on the next page. The first bug has been found for you and is listed as an example in the table.

```
1   public class Deal {
2       public static void main(String[] args) {
3           int CARDS_PER_PLAYER = 5
4
5           // number of players
6           int numPlayers = Integer.parseInt(args[1]);
7
8           String[] suit = { "Clubs", "Diamonds", "Hearts", "Spades" };
9           String[] rank = { "2", "3", "4", "5", "6", "7", "8", "9",
10                            "10", "Jack", "Queen", "King", "Ace" };
11
12          // avoid hardwired constants
13          int numSuits = suit.length;
14          int numRanks = rank.length;
15          int numCards = numSuits * numRanks;
16
17          if (CARDS_PER_PLAYER * numPlayers > numCards) {
18              System.err.println("Too many players");
19              System.exit(1);
20          }
21
22          // initialize deck with all combinations of ranks and suits
23          String[] deck = String[numCards];
24          int r = 0;
25          for (int i = 0; i < numRanks; i++) {
26              for (int j = 0; j < numSuits; j++) {
27                  deck[numSuits * i + j] = rank(i) + " of " + suit[j];
28              }
29          }
30
31          // shuffle the deck
32          for (int i = 0; i <= numCards; i++) {
33              int r = i + (Math.random() * (numCards - i));
34              String t = deck[i]; // swap the cards
35              deck[r] = deck[i];  // at indices i and r
36              deck[i] = t;        // in the deck
37          }
38
39          // print shuffled deck
40          for (int i = 0; i < PLAYERS * CARDS_PER_PLAYER; i++) {
41              System.out.println(deck[i]);
42              if (i % CARDS_PER_PLAYER == CARDS_PER_PLAYER - 1)
43                 System.out.println();
44          }
45
46  }
```

5.1. (10 pts) List the 7 errors (excluding the given example) that will cause the code to fail to compile or cause a runtime exception, and 1 logical bug in the code that will cause the program to not perform as expected. Each correction requires one line of code or less. **Use the last row for the logical bug**.

Line Number	**Error**	**Correction**
3	Missing Semicolon	`int CARDS_PER_PLAYER = 5;`
	<u>Logical Bug:</u>	

SECTION 6: TRACERY (10 pts)

Trace through the following code. Assume that the program is executed using the command:

```
java HofstadterConway 5
```

Whenever you reach a `System.out.println()` statement, write the output that would be printed as a new row in the table. You may not need to use all of the table's rows.

```
public class HofstadterConway {
      public static int HC(int n) {
            if(n == 1) return 1;
            if(n == 2) return 1;
            int a = HC(HC(n-1)) + HC(n – HC(n – 1));
            System.out.println(n + " : " + a);
            return a;
      }

      public static void main (String[] args) {
            int num = Integer.parseInt(args[0]);

            System.out.println("Answer: " + HC(num));
      }
}
```


BACKGROUND (Note: Not needed to answer the question above!)

This function generates the **Hofstadter – Conway $10,000 Sequence!** It acquired its name because John Horton Conway offered a prize of $10,000 to anyone who could find the first position, p in the sequence for which |a(n)/n – ½| < 1/20 for all n > p. The prize was subsequently claimed by Mallows, who found that p = 1,489.

It was later found that Hofstadter had found the sequence and its structure 10 - 15 years before Conway posed his challenge.

SECTION 7: SORTING (8 pts total)

Assume we are given the array: `{8, 7, 3, 0, 9, 2, 6}`.

Recall that both insertion sort and selection sort maintain a separation of the array into two parts: a left part that is kept in sorted order and a right part that is unsorted. Each step of the algorithm, one additional element is added to the left (sorted) portion.

For example, in insertion sort, 8 is initially marked as sorted: `{8, 7, 3, 0, 9, 2, 6}`. After the first step of sorting, 7 is added to the sorted section, thus making the state of the array: `{7, 8, 3, 0, 9, 2, 6}`, where the sorted numbers are underlined.

7.1. (2 pts) What would be the state of the array after the third step of insertion sort?

A. `{0, 2, 3, 8, 9, 7, 6}`
B. `{0, 2, 3, 6, 9, 7, 8}`
C. `{0, 2, 3, 7, 9, 6, 8}`
D. `{0, 3, 6, 8, 9, 2, 7}`
E. `{0, 3, 7, 8, 9, 2, 6}`

7.2. (2 pts) What would be the state of the array after the third step of selection sort?

A. `{0, 2, 3, 8, 9, 7, 6}`
B. `{0, 2, 3, 6, 9, 7, 8}`
C. `{0, 2, 3, 7, 9, 6, 8}`
D. `{0, 3, 6, 8, 9, 2, 7}`
E. `{0, 3, 7, 8, 9, 2, 6}`

For the following questions, a swap occurs when the sorting algorithm switches the position of two elements. A comparison is defined as a check between two elements.

For example, in the first step of insertion sort, both 7 and 8 are compared and swapped.

7.3. (2 pts) How many times is 8 swapped by the end of selection sort?

A. 2 C. 3 E. None of the above
B. 0 D. 4

7.4. (2 pts) How many elements are compared with 3 by the end of insertion sort?

A. 1 C. 0 E. None of the above
B. 2 D. 4

Since there are a number of people who still need to take the makeup midterm, please DO NOT discuss the exam or post anything about it on Piazza. Thank you! That's it!!

SCRAP PAPER

(This page is intentionally blank)

ANSWER KEY

SECTION 1: MISCELLANEOUS

1.1. (2 pts) `B`

1.2. (2 pts) C

1.3. (2 pts) D

1.4. (2 pts) B

SECTION 2: CONDITIONALS AND LOOPS

2.1. Blank A: `n <= 27 OR n < 81`

Blank B: `n * 3`

2.2.

i. B **(2 pts)**

ii. C **(2 pts)**

iii. None **(2 pts)**

iv. D **(2 pts)**

SECTION 7: SORTING

7.1. `E` **(2 pts)**

7.2. `A` **(2 pts)**

7.3. `C` **(2 pts)**

7.4. `E` (5 times) **(2 pts)**

SECTION 3: VARIABLES, OPERATORS, TYPES

3.1.

Code	DataType/Error	Value of z/reason
`int x = 5;` `??? z = x / 2;`	int or double	z = 2 if int z = 2.0 if double

`int x = 0;`
`int y = 1;`
`??? z = (x = y);`
	int	z = 1

`String x = "CIS";` `??? z = x + 1 + 1 + 0;`	String	z = "CIS110"
`??? z =` `Double.parseDouble(2.0);`	ERROR	parseDouble converts String to double. Argument should be a string.
`int[] x = {2, 5, 1, 0};` `??? z = x[x[0]];`	int	z = 1
`String x = "Trump vs Sanders vs Clinton";` `??? z = x.charAt(12);`	char	z = 'd'
`??? z = 'a' < 'b';`	boolean	z = true

SECTION 4: FUNCTIONS!

4.1. **Example full credit answers** (using division and mod, without a loop)

```
public static int mutantBunnyEars(int numBunnies){
     // count pairs of bunnies, which have 5 ears total
     int answer = 5 * numBunnies / 2;
     // add on 3 ears for the odd bunny
     answer += 3 * numBunnies % 2;
     return answer;
}

public static int mutantBunnyEars(int numBunnies){
     return 5 * numBunnies / 2 + 3 * numBunnies % 2;
}
```

Partial credit response

```
public static int mutantBunnyEars(int numBunnies){
        int n = 0;
        int i = 1;

        while(i <= numBunnies){
             if (i % 2 == 0) n += 2;
             else            n += 3;
             i++;
        }
        return n;

     }
```

4.2

```
 public static int numIncreases (int[] arr) {
     if (arr == null) return 0;

      int sum = 0;
      for (int i = 1; i < arr.length; i++) {
          if (arr[i-1] < arr[i]) {
             sum++;
          }
      }
      return sum;
   }
```

SECTION 5: DEBUGGING

5.1. You had to find 7 out of the 8 possible syntax errors, plus the logical error

Line Number	Error	Correction
3	Missing Semicolon	`int CARDS_PER_PLAYER = 5;`
6	Incorrect index to args. args[0] contains the number of players	`int PLAYERS = Integer.parseInt(args[0]);`
23	Array initialization error. Missing keyword new	`String[] deck = new String[CARDS];`
27	Incorrect way of accessing array element. Change `rank(i)` to `rank[i]`	`deck[SUITS*i + j] = rank[i] + " of " + suit[j];`
32	Loop exceeds array bounds.	`for (int i = 0; i < CARDS; i++) {`
33	Extra declaration of r	Delete `int`
33	Missing cast to int	`r = i + (int)(Math.random() * (CARDS-i));`
40	Undefined variable `PLAYERS`	`Change to numPlayers`
45	Missing closing bracket for main	`Add }`

34	**Logical Error:**
	Value of `deck[r]` is lost in swap.
		`String t = deck[r];`
		`deck[r] = deck[i];`
		`deck[i] = t;`

SECTION 6: TRACERY (Hofstadter – Conway $10,000 Sequence!) (10 pts)

3 : 2
3 : 2
4 : 2
3 : 2
3 : 2
4 : 2
3 : 2
5 : 3
Answer: 3