CIS 110 — Introduction to Computer Programming
Summer 2018 — Midterm

Name: _____

Recitation ROOM : _____

Pennkey (e.g., paulmcb): _____

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 90 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 <u>any</u> 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 code must be written out in proper java format, including all curly braces and semicolons.
- **<u>Do not separate the pages.</u>** You may tear off the one scratch page at the end of the exam. This scratch paper must be turned in or you lose 5 points.
- Turn in 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. **Only answers on the FRONT of pages will be grading. The back is for scratch work only.**
- 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 <u>immediately</u>.**
- We wish you the best of luck.

Scores: [For instructor use only]

Question 1		4 pts
Question 2		20 pts
Question 3		6 pts
Question 4		6 pts
Question 5		6 pts
Question 6		10 pts
Question 7		10 pts
Total:		**62 pts**

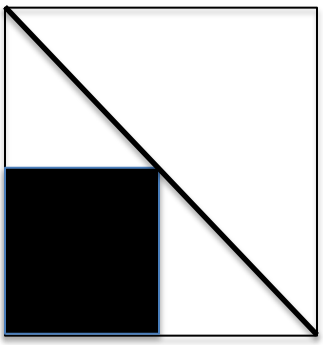
1.) Miscellaneous (5 points)

1.1) (1 point) The Easy One:

- Check that your exam has all 9 pages (excluding the cover sheet, which is no numbered).
- 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.

1.2) (2 points) In the space provided, write code to draw the figure pictured in the window to the right (you can assume the square is half as wide as the window it is in. You do not have to draw the window, only the shapes.

```
PennDraw.line(0.0, 1.0, 1.0, 0.0);
PennDraw.filledRectangle(0.25, 0.25, 0.25, 0.25);
```



1.3) (1 point) How many times would the following print HelloWorld?

```
int i = 17;
while (i >= 0) {
    i /= 2;
    if (i % 2 == 0) {
       System.out.println("Hello World");
    }
}
```

a) 0

b) 1

c) 5

d) 6

e) Infinite (infinite loop)

f) 0, but with an infinite loop

Note: while f was the intended answer, we accepted 0 because there were unfixed typos on the original exam that would have prevented compilation.

2.) Operators and Expressions (20 points) – 1 point for each blank

For each code fragment, (a) fill in the most appropriate word in the 1st column(if there are multiple blanks, all blanks are the same word) and (b) 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 (you do not need to write the exact error message, just a general explanation). The first two problems have been completed for you.

DO NOT WRITE WHAT PRINTS. ONLY THE TYPE IN THE BLANK AND THE VALUE OF Z

`____ z = 11;` `z++;`	int	12
`int x = "Hello World";` `____ z = x.length();`	ERROR	Cannot set an int variable to a String

`String str = "25.0";`
 	
`int z = ___.parseDouble(str);`
	ERROR	Double.parseDouble() would return a double, so this wouldn't compile

`String s = "Catdog";`
 	
`____ z = s.charAt(4);`
	char	'o'

`____ z = {'a', 'b', 'c', 'd'};`
 	
`int x = z[2];`
 	
`z[2]++;`
	char[]	{'a','b','d','d'}

`double x = Math.random();`
`if (x > 0.5) {`
`____ z = "TAILS";`
`}`
`System.out.println(z == "TAILS");`
	ERROR	z would be out of scope outside the if statement, code wouldn't compile

`____ z = 3 / 2 + 1.5 * 2`	double	4.0

`____[] x = new ____[4];`
 	
`x[3] = 'c';`
 	
`____ z = x[0];`
	char	0 (or the null character, '/0')

(continued on the next page)

???? z = {"cat", 5, true}; ???? x = z; x[0] = "steve";	ERROR	No data type can hold these values
???? z = {"cat", "5", "true"}; ???? x = z; x[1] = "steve";	String[]	{"cat","steve","true"}
String x = "" + 1 + 2 ???? z = x.equals(3);	ERROR	Cannot call the string equals function with an int argument.
String x = 1 + 2 + ""; ???? z = "1" + '2'; System.out.println(z == x);	String	"12"

3) Conditionals (6 points total)

Fill in the blanks for the following method that takes in 3 integers and returns the smallest number. You cannot change any existing code.
Note: there was confusion caused by this question, where the function was called max, but the behavior was min. Therefore, I accepted both min or max implementations. The answer below illustrates min

```
public static double min(double a, double b, double c) {

    if (b < c) { //2 points for this blank

        if (b < a) { //2 points blank
            return b;
        } else {
            return a;
        }
    } else {
        if (a > c) {

            return c; //1 point
        } else {

            Return a; //1 point
        }
    }
}
```

4) (6 points) Tracing

In a failed attempt to make computer science fun, Will wrote the following function. Trace the function using the inputs below to determine the output. You are encouraged to use scratch paper to help you trace

```
1   public static int fun(int decorations, int surprise) {
2       int party = 0;
3       int cake = -1 * surprise;
4       while (surprise > cake) {
5           surprise--;
6           for (int hats = decorations; hats > 0; hats -= 2) {
7              party++;
8              hats++;
9           }
10          surprise--;
11      }
12
        return party;
}
```

ONLY WRITE WHAT RETURNS IN THE BOX (1.5 points each)

A) `fun(1,3);`

3

B) `fun(2,3);`

6

C) `fun(4,3);`

12

D) `fun(12, 10);` *(tip: look at the relationship between input and output on the last)*

120

5) Method Writing – Moderate (6 points)

You will be writing an **int** method that takes in an integer array (int[] arr) and one int number (int c). This function returns the number of times c appears in arr.

You can ASSUME the following:

- The array has at least 1 element (is not empty)
- The number c could appear as few as zero times, and as many as every time.

Do not write any code to address those assumptions, just assume those things are all true.

```
public static int count(int[] arr, int c) {

        int count = 0;

        for(int i = 0; i < arr.length; i++) {

            if (arr[i] == c) {
                    count++;
            }
        return count;
}
```

6) Debugging (10 points)

The function below is supposed to give the nth letter of the alphabet, where 0 is 'A', 1 is 'B', and so on. In the code below, there are 6 bugs that prevent the code from COMPILING (only worry about compile time errors, not logic bugs). Identify these bugs by the LINE number, then suggest a fix to the line of code. Example

Bug 1: Line 1 – no return type on function header.
Should be "public static **char** getNthLetter(int n)…

That's 1 down, 5 more for you to find! Note that you are only allowed to edit the line of code you say would cause a compile time error. You are not allowed to change any other lines of code;

There were actually 7 bugs here, people got full credit if they found at least 6. The bugs are highlighted in red

```
1  public static getNthLetter (int n) {
2      char ch = "A"; //should be 'a'
3      //if n is less than 0 or greater than 25, return
4      / a whitespace character //you need two slashes
5      if (n < 0 or n > 25) { //should be ||
6          return ' ' //no semi colon
7      }
8      for (int counter = 0, counter < n, counter++) { //; not ,
9          int newChar = ch;
10         newChar++;
11         ch = newChar; //need to typecast to (char)
12     }
13     return ch;
14 }
```

Here is an image of the compilable code

```
public static char getNthLetter (int n) {
    char ch = 'A'; //should be 'a'
    //if n is less than 0 or greater than 25, return
    // a whitespace character //you need two slashes
    if (n < 0 || n > 25) { //should be ||
        return ' '; //no semi colon
    }
    for (int counter = 0; counter < n; counter++) { //; not ,
        int newChar = ch;
        newChar++;
        ch = (char) newChar; //need to typecast to (char)
    }
    return ch;
}
```

7) Method Writing – Harder (10 points)

An anagram is when two words have the same letters, but in a different order. For example, "binary" and "brainy" are anagrams. Is this question, you will write the method **isAnagram** that takes in two strings and determines whether or not they are anagrams. Use an appropriate return type for what this method is asking you to do.

You can assume (do not check) that all characters in the input strings are **lowercase.**

Be careful, however, as this is harder than it first looks. Example: "need" and "eden" are anagrams, but "den" is not an anagram of either because it doesn't have two e's. Further, neither is "nedd" as while it has the same letters, it only has 1 e and 2 d's.

```
There were a number of ways to do this, below is in my opinion the
easiest.
public static boolean isAnagram(String a, String b) {

      //If the strings aren't the same length,
      //they cannot be anagrams
      if (a.length() != b.length) {
            return false;
      }

      //get letter counts of both strings
      int[] aLetterCounts = new int[26];
      int[] bLetterCounts = new int[26];

      //only need one loop since both strings have
      //same number of letters
      for (int i = 0; i < a.length; i++) {
            char aChar = a.charAt(i);
            char bChar = b.charAt(i);

            //'a' - 'a' is 0, 'b' - 'a' is 1 ...
           //'y' - 'a' is 24 'z' - 'a' is 25
           //we use this to keep track of letter counts
           aLetterCounts[aChar - 'a']++;
           bLetterCounts[bChar - 'a']++;
      }

      //if any letter counts don't match
      //return false. Otherwise they are anagrams
      for (int i = 0; i < aLetterCounts.length; i++) {
            if (aLetterCounts[i] != bLetterCounts[b]) {
                  return false;
            }
      }
      //if you made it this far, they are anagrams
      return true;
}
```

Extra Space for Answers

DO NOT RIP THIS PAGE OFF. ANY WRITING ON THIS PAGE CAN BE GRADED

Scratch Paper

YOU MAY USE THIS PAGE FOR SCRATCH WORK, BUT NOTHING ON IT WILL BE GRADED