CIS 110 — Spring 2022 — Exam 1

Full Name: _____

Recitation #: _____

Pennkey (e.g. sharry): _____

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 are below. Not complying will lead to a 0% score on the exam.

- Do not open this exam until told by the proctor.
- You will have exactly 110 minutes to take this exam.
- Make sure your phone is turned OFF (not on vibrate!) before the exam starts.
- Fill out the information and declaration above. Write your Penn ID (the numbers!!!), eg. 12345678, on every page in the space given before you work on anything. If you don't do so, we won't be able to find your answers and you won't get the points you deserve.
- Food, gum, and drink are strictly forbidden. Masks are mandatory. Can drink water through a straw.
- Green PennOpen passes required. If you have a non-compliance Red Pass, you cannot take the exam and will get a 0 (no makeup).
- 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 code must be written in proper java format, including all curly braces and semicolons.
- Do not separate the exam pages. Do not take any exam pages with you. The entire exam packet must be turned in as is.
- Only answers on the FRONT of pages will be graded. There are two blank pages at the end of the exam if you need extra space for any graded answers. You may use the back of pages for additional scratch work.
- 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 you.
- When you turn in your exam, you may be required to show your PennCard. If you forgot to bring your ID, talk to an exam proctor immediately.

Q1	Q2	Q3	Q4	Q5
28 pts	18 pts	15 pts	19 pts	30 pts

Q1. Short & Sweet (14 Questions, 2 pts each)

All your answers must be placed in the Answer blank below each question. Only answers placed in the designated area will be graded.

Question 1.1:

True or False? The following code compiles.

```
String myID = "patrick" + 3497;
```

Answer

☑ True ☐ False

Question 1.2:

True or False? Binary search can be used to search for numbers in an array that has duplicates.

Answer

☑ True ☐ False

Question 1.3:

True or False? The following test case passes. (Assume it's written a valid file with all necessary imports and valid syntax outside of this snippet of code.)

```
@Test
public void test() {
    assertEquals(5.2, 5.1, 0.01);
}
```

Answer

☐ True ☑ False

Question 1.4:

True or False? All PennDraw functions have the return type void.

Answer

☐ True ☑ False

Question 1.5:

True or False? The following code will NOT print 3.

```
int[] arr = {5, 3, -1};
if (true) {
    System.out.println("CIS");
} else {
    System.out.println(arr.length);
}
```

Answer

☑ True ☐ False

Question 1.6:

True or False? The variable initialized in the first portion of a for loop must be an `int`.

Answer

☐ True ☑ False

Question 1.7:

True or False? The "`return`" keyword must be followed by a variable.

Answer

☐ True ☑ False

Question 1.8:

Fill in the blank with a value that makes the following code ONLY print "`I love CIS 110`".

```
String str = ____________;
if (str.charAt(2) <= 'b') {
    System.out.println("I");
}
```

```
if (str.charAt(2) <= 'd') {
    System.out.println("I love");
}
if (str.charAt(2) < 'f') {
    System.out.println("I love CIS 110");
}
```

Answer: **Any string that has an 'e' as its third character (at index 2)**

Question 1.9:

Fill in the blank with the conditional operator (&&, ||) that would make `myBool` be `true`

```
boolean x = true;
boolean y = false;
boolean myBool = !(!(x || y) ____ (!y && x));
```

Answer: **&&**

Question 1.10:

Fill in the blank with `variable`'s correct type based on its value. If there's an error, indicate what error.

```
____ variable = 7 > 4.0;
```

Answer: **boolean / Boolean / bool (boolean is the most accurate answer)**

Question 1.11:

What will the following block of code print?

```
boolean x = true;
boolean y = false;
if (x && y) {
    System.out.println("1");
}
if (!y) {
    System.out.println("2");
} else if (x || y) {
    System.out.println("3");
    if (x && !y) {
        System.out.println("4");
```

```
    }
}
```

Answer: **2 (“2” will also be accepted)**

Question 1.12:

What will the following block of code print?

```
public static void main(String[] args) {
    int a = 0;
    int b = 7;
    double c = 0.4;
    if (a <= b) {
        c = 4.2;
    }
    if (a > c) {
        a = 7;
    } else if (b > c) {
        b = a;
        a = 4;
    }
    System.out.println(a + b + c);
}
```

Answer: **8.2**

Question 1.13:

What will be printed by the following code?

```
int x = 71 % 5;
int y = x;
if (x < 10) {
    x = x + 10;
}
if (x % 10 == y) {
    System.out.println(x - 1);
} else {
    System.out.println(x + 1);
}
```

a) 10

b) 11
c) 12
d) 1
e) 12

Answer (Select one - A, B, C, D, E):
☑ A ☐ B ☐ C ☐ D ☐ E

Question 1.14:

Which is not a valid way of method overloading?

a)
```
public static double add(int x, int y)
public static double add(int x, int y, int z)
```
b)
```
public static double add(int x, int y)
public static double add(int x, double y)
```
c)
```
public static double add(int x, double y)
public static double add (double x, int y)
```
d)
```
public static double add(int x, int y)
public static int add(int x, int y)
```

Answer (Select one - A, B, C, D, E):
☐ A ☐ B ☐ C ☑ D

Q2. Long Fill In The Blank - Naming Names (18 pts, 2 per blank)

Harry wants to come up with characters for a graphic novel he's writing. To follow the tradition of superhero and supervillain names being alliterative (both names starting with the same letter: Green Goblin, Bruce Banner, etc.), he wants to write a program that will find alliterative pairs of adjectives and nouns for him. Given a `String[]` of adjectives and a `String[]` of nouns, help him write a function that will return a `String[]` of all alliterative pairs **with a space added between words**.

For example,

```
String[] adjectives = {"Green", "Grand", "Hungry", "Red"};
String[] nouns = {"Goblin", "Hippo", "Dog"};
findAlliteration(adjectives, nouns);
```

should evaluate to {"Green Goblin", "Grand Goblin", "Hungry Hippo"}. Notice the spaces added between words.

Please fill in the corresponding code (displayed on the next page) for each blank denoted by a ___# ___. You need to fill in the blanks and put your answers in the table on the page immediately after the code. Only the answers in the box will be graded.

Code (you are allowed to write in this box - none of your marks here will be graded) :

```
public static String[] findAlliteration(String[] adjectives, String[] nouns) {
    // count how many alliterative pairs we have
    int pairsCount = __0__;
    int numAdjectives = adjectives.length;
    int numNouns = nouns.length;
    for (int i = 0; i < __1__; i++) {
        for (int j = 0; j < numNouns; j++) {
            // get the first char in both the current strings we're looking at.
            char firstAdjLetter = __2__; // pay attention to the types!
            char firstNounLetter = __3__;
            if (firstAdjLetter == firstNounLetter) {
                __4__;
            }
        }
    }

    // initialize the array with enough space for all pairs
    int currentIdx = 0;
    String[] pairs = new __5__;

    // fill in the array, generating the pairs again
    for (int k = 0; k < numNouns; k++) {
        for (int l = 0; l < numAdjectives; l++) {
            // not the same as before!
            String firstAdjLetter = adjectives[l].substring(0, 1);
            String firstNounLetter = nouns[k].substring(0, 1);
            if (__6__) {
                String pair = adjectives[l] + __7__;
                pairs[currentIdx] = pair;
                currentIdx = __8__;
            }
        }
    }
    return pairs;
}
```

Write your answers here:

Blank Number	Your Answer
0	0
1	numAdjectives
2	adjectives[i].charAt(0);
3	nouns[j].charAt(0);
4	pairsCount++;
5	String[pairsCount];
6	firstAdjLetter.equals(firstNounLetter)
7	nouns[k]
8	currentIdx + 1;

Q3. Recursive Tracing (15 pts)

Dustin is a big fan of Elon Musk. More importantly, he loves the innovation he exhibited in naming his child X Æ A-12. He's been thinking of other cool ways to come up with interesting names. Help Dustin trace through this `mystery()` method and **write what would be printed to the console for the two function calls below**. We have given you the first one as an example. Write what prints in the box underneath each question.

`Example: mystery("X", "AEA", 12)` would print `XAEA-12`

```
public static void mystery(String s1, String s2, int a) {
    System.out.println(s1 + s2 + "-" + a);
    if (a == 0 || a == 12) {
        return;
    } else if (a % 3 == 0 && a % 2 == 0) {
        mystery(s1 + s1, s2 + s2, a - 1);
    } else if (a % 3 == 2) {
        mystery(s1 + s1, s2, a - 1);
    } else if (a % 3 == 1) {
        mystery(s1, s2 + s2, a - 1);
    } else {
        return;
    }
}
```

Question 3.1: `mystery("Ri", "A$ap", 10);`

1. `RiA$ap-10`
2. `RiA$apA$ap-9`

Question 3.2: `mystery("Kim", "Ye", 5);`

1. `KimYe-5`
2. `KimKimYe-4`
3. `KimKimYeYe-3`

Q4. Short Coding - Surprise [Ci]pher[s] (19 pts)

TAs Becca and Alan are planning a secret sendoff for the Senior TAs who will be graduating. Since all the TAs are on the same Slack workspace, they have to get creative so that the surprise stays a surprise! They have decided to hide their messages in some text excerpts and an array of integers, which are a series of character indices for extracting characters from the text. Once the other TAs have this information, they want your help in writing a function to extract the hidden message.

This function `public static String decodeText(String text, int[] indices)` will output a `String message` when given a text in the form of a `String`, and an `int[] array` of indices to extract. You should extract characters in the order they appear in the array, from left to right.

To make this more interesting, the arrays may contain negative numbers and numbers that are too big! If a number given in the indices array is negative, you simply skip/disregard it. If it's too big (such that it's out of bounds), the index wraps around.

Example:

```
String text = "My friend, how are you today?";
int[] indices = {0, 6, -1, 35, 23, 2, 15, 23, 2, 3, 12, -7, 21, 62};
System.out.println(decodeText(text, indices)); // prints "Meet at four"
```

This table has all the indices for each character in the string laid out for you to understand the example above.

M	y		f	r	i	e	n	d	,		h	o	w		a	r	e		y	o	u		t	o	d	a	y	?
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28

Notice here that values -1 and -7 in `int[] indices` are ignored because they are negative indices. Values like 35 and 62 are considered out of bounds, but wrap around in a way that 35 represents the `'e'` at index 6 and 62 represents the `'r'` at index 4. (The length of the text is 29).

Before you write the function, complete the JUnit test below so that it tests the example described above. That is, write a JUnit test given the input `String text` and `int[] indices` from above (think about how you would write a JUnit test that tests `decodeText` on those inputs)! You may assume that the function you are writing is in a file called `Exam.java` and your test case is written in a file called `ExamTest.java` which has all the necessary import statements already in it.

After completing the test case, please fill in the function as described above!

Question 4.1: Complete the test below!

```
@Test
public void testDecode() {
    String text = "My friend, how are you today?";
    int[] indices =  {0, 6, -1, 35, 23, 2, 15, 23, 2, 3, 12, -7, 21, 62}

    // fill in the rest of this test

    String expected = "Meet at four";
    String actual = Exam.decodeText(message, indices);
    assertEquals(expected, actual);


}
```

Question 4.2: Complete the function below!

```
public static String decodeText(String text, int[] indices) {
    // implement this function!

    String out = "";
    for (int i = 0;  i < indices.length; i++) {
        int index = indices[i];
        if (index >= 0) {
            index = index % text.length();
            out += text.charAt(index);
        }
    }
    return out;









}
```

Q5. Long Coding - CIS 110 Wordle

Wordle is a game where you must discover what the secret 5-letter word is. In order to discover the word, you must guess what the word is; upon guessing a word, Wordle will provide you feedback on every letter in your guessed word. For every letter in the word you guessed, Wordle will tell you if that letter is either (a) not in the secret word, (b) in the secret word at the same index that it's currently at in the word you guessed, or (c) in the secret word at a different index.

TAs (and Wordle enthusiasts) Shivin and Gian are trying to write some code to help narrow down their guesses as they play Wordle - and they need your help to do so!

They are making some simplifying assumptions:

- assume all words are written in uppercase characters only.
- assume that no words have repeating characters (i.e. each 5-letter secret word as well as each guessed word has 5 distinct/different characters - words like "LEVER" and "APPLE" are neither valid secret words nor guesses).

`Given Function: scoreWord`

As mentioned above, Wordle scores each guessed word through the underlined process above. We have provided you a function `scoreWord` which will take in a guess word and output your guess word's score as an integer array with integer scores for each letter in your guess word.

For any letter *ch* at an index *i* in the guessed word, the array at index *i* will store a -1 if *ch* is not in secret word, a 0 if *ch* is in the secret word at a different index than *i*, or a 1 if *ch* is in the secret word at index *i.* Below are some examples of how the function works:

- If the secret word is "HANDY" and the guess is "FIRES", the call scoreWord("FIRES") will return [-1, -1, -1, -1, -1]. **None of the letters in "FIRES" are in the secret word.**
- If the secret word is "HANDY" and the guess is "FLARE", the call scoreWord("FLARE") will return [-1, -1, 0, -1, -1]. **Letter A is in the secret word, but not at index 3.**
- If the secret word is "EQUAL" and the guess is "PETAL", the call scoreWord("PETAL") will return [-1, 0, -1, 1, 1]. **Letter's A, E, and L are in the secret word, but E is in the wrong place.**

```
/* Description: This function scores a guess word
 * Input: The word you are guessing → This is a 5 letter word.
 * Output: An 5-element array filled with 1s, 0s, and -1s.
 * If you're a Wordle player: -1 means Gray, 0 means Yellow, 1 means Green
 * Note: YOU DO NOT HAVE TO IMPLEMENT THIS FUNCTION. WE ARE PRETENDING IT EXISTS
 * AND IS FUNCTIONAL.
 */
public static int[] scoreWord(String guessWord);
```

`Question 5.1:` `Determining if a word is possible. (18 pts)`

You solve the Wordle by using the "score" on one guess to educate your future guesses. To mirror this in Java, **write a function called `isWordPossible`**. The function should take in a `firstGuess` word and a `nextGuess` word. Based on Wordle's score of the `firstGuess` word, you should determine if the `nextGuess` word could possibly be the secret word. Below are some examples of how the function works, assuming that the secret word is "HANDY":

- If `firstGuess` is "FIRES", and `nextGuess` is "FLARE", the function should return ***false***. Calling `scoreWord("FIRES")` returns [-1, -1, -1, -1, -1] since there is no overlap between "FIRES" and "HANDY". Since "F", "E" and "R" are not in the secret word, the `nextGuess` word "FLARE" could not possibly be the secret word (since it contains an "F" and an "E" and an "R").

- If `firstGuess` is "FLARE" and `nextGuess` is "BATON", the function should return ***true***. Calling scoreWord("FLARE") returns [-1, -1, 0, -1, -1], showing that "A" is in the secret word at a location other than index 2, and that the secret word does not have an "F", "L", "R", or "E". A `nextGuess` of "BATON" meets all of these constraints (no "F", "L", "R", or "E", and has the letter "A" at an index other than 2). Therefore, it is possible that `nextGuess` could be the secret word.

Hint: You should use the given function `scoreWord` from above. You may also use the `str1.contains(ch)` method, which returns true if `char ch` is in `str1`. For example, "FLARE".contains('Z') returns false and "FLARE".contains('A') returns true.

<u>How to Write this Function</u>: Your function should determine if the candidate word is possible by following the below steps:

- You should first get the score of your `firstGuess`
- For each letter's individual integer score, you should check the following:
    - If the score is -1 at an index, the letter in your `firstGuess` at this index should NOT be a letter in your `nextGuess`
    - If the score is 0 at an index, the letter in your `firstGuess` at this index should be a letter in your `nextGuess`, but NOT at this same index
    - If the score is 1 at an index, the letter in your `firstGuess` at this index should match the letter at this index in `nextGuess`
- If these conditions are met for every letter, then `nextGuess` could potentially be the secret word.

Question 5.1: Write your code below!

```
/* Description: Determines if based on a firstGuess's score whether or not
 * the nextGuess could possibly be the secret word.
 *
 * Inputs: your firstGuess and your nextGuess (after the firstGuess)
 * Output: Boolean - whether or not nextGuess could be the secret word
 */
public static boolean isWordPossible(String firstGuess, String nextGuess) {
    // implement this function!
    int[] score = scoreWord(firstGuess);
    for (int i = 0; i < 5; i++) {
        if (score[i] == -1 && nextGuess.contains(firstGuess.charAt(i))) {
            return false;
        }
        if (score[i] == 0 && !(nextGuess.contains(firstGuess.charAt(i)) &&
            firstGuess.charAt(i) != nextGuess.charAt(i))) {
            return false;
        }
        if (score[i] == 1 && nextGuess.charAt(i) != firstGuess.charAt(i)) {
            return false;
        }
    }
    return true;









}
```

Question 5.2: Finding all valid 'candidates' (12 pts)

Now we will write a function that counts and prints all possible words the secret word could be. Given a `guess` word, as well the collection of all words in Wordle's Dictionary (a String array), you must print all possible words that the secret word can be. To do this, you should determine which of the words in the dictionary are possibly the secret word (via your function from part 1) and print them each on their own line. Then, you should return the number of words you printed out. ***You should assume that your function from 5.1 is working correctly and that you can use it here.***

Question 5.2: Write your code below!

```
/* Description: Given your guess word, as well as the collection of all
 * words in Wordle's Dictionary, return all possible words that the Wordle
 * Secret Word can still be.
 *
 * Inputs: A word you'll guess, and String[] of all possible wordle words
 * Output: Return the number of remaining words the secret word can still
*  be. Additionally, print out all those remaining words.
 */
public static int printAndCountValidGuesses (String guess,
                                              String[] wordleDict) {
    // implement this function!

    // init var to count the number of candidates
    int count = 0;

    // now print them all
    for (int i = 0; i < wordleDict.length; i++) {
        if (isWordPossible(guess, wordleDict[i])) {
            System.out.println(wordleDict[i]);
            count++; //increment the count
        }
    }

    // return the count
    return count;









}
```

Extra Answers Page (This page is intentionally blank)

You may use this page for additional space for answers; keep it attached to this exam. Clearly note on the original question page that your answer is on this extra page, and clearly note on this page what question you are answering.

Extra Answers Page (This page is intentionally blank)

You may use this page for additional space for answers; keep it attached to this exam. Clearly note on the original question page that your answer is on this extra page, and clearly note on this page what question you are answering.