CSE 121 WINTER 2024 FINAL EXAM MARCH 12, 2024

Name: _____

Section: _____ Student **Number** (not UWNetID): _____

Rules/Guidelines:

- You must not begin working before time begins, and you must stop working **promptly** when time is called. Any modifications to your exam (writing or erasing) before time begins or after time is called will result in a penalty.
- You are allowed one page of notes, no larger than 8.5 x 11 inches. You may not access any other resources or use any electronic devices (including calculators, phones, or smart watches, among others) during the exam. Using unauthorized resources or devices will result in a penalty.
- In general, you are limited to Java concepts or syntax covered in class. You may not use `break`, `continue`, a return from a `void` method, `try`/`catch`, or Java 8 features.
- You are limited to the standard Java classes and methods listed on the provided reference sheet.
- You do not need to write import statements.
- If you abandon one answer and write another, **clearly cross out** the answer(s) you do not want graded and **draw a circle or box around the answer you do want graded**. When in doubt, we will grade the answer that appears in the space indicated, and the first such answer if there is more than one.
- If you require scratch paper, raise your hand and we will bring some to you. If you write an answer on scratch paper, **please write your name and clearly label** which question you are answering on the scratch paper, and **clearly indicate** on the question page that your answer is on scratch paper. Staple all scratch paper you want graded to the **end** of the exam before turning in.
- Answers must be written as proper Java code. Pseudocode or comments will not be graded. However, the exam is not graded on code quality. You are not required to include comments.
- You are also allowed to abbreviate "`System.out.print`" and "`System.out.println`" as "S.o.p" and "S.o.pln" respectively. You may **NOT** use any other abbreviations.

Grading:

- Each problem will receive a single E/S/N grade.
- On problems 1 through 3, earning an E requires answering all parts correctly and earning an S requires answering almost all parts correctly.
- On problems 4 through 6, earning an E requires an implementation that meets all stated requirements and behaves exactly correctly in all cases. Earning an S requires an implementation that meets all stated requirements and behaves exactly correctly in most cases or behaves nearly correctly in all cases.
- Minor syntax errors will be ignored as long as it is unambiguous what was intended (e.g. forgetting a semicolon, misspelling a variable name where there is only one close option). Major syntax errors, or errors where it is unclear what was intended, may have an impact on your grade.

Advice:

- Read all questions carefully. Be sure you understand the question before you begin your answer.
- The questions are not necessarily in order of difficulty. Feel free to skip around. Be sure you are able to at least attempt every question.
- Write clearly and legibly. We cannot award credit for answers we cannot read.
- If you have questions, raise your hand to ask. The worst that can happen is we will say "I can't answer that."
- Ask questions as soon as you have them. Do not wait until you have several questions.

Initial here to indicate you have read and agree to these rules:

Name: _____

1. Code Comprehension

(a) Trace the evaluation of the following expressions, and give their resulting values. Make sure to give a value of the appropriate type. (i.e. Be sure to include a `.0` at the end of a `double` value, or `""` around `String`s.) Write your answer in the line to the right of each expression.

i. `(30 - 27) + 8 / 2 / 5 % 3`

ii. `12 / (5.0 - 1) + 3 % 2 + "g"`

iii. `!(12 / 3 <= 10.0 % 2) && 5.0 % 2 == 1`

(b) Consider the following code:

```
public static int m(int x, int y) {
    while (x > 0) {
        x = x - y;
        y = y + x;
        System.out.print(y + " ");
    }
    return x;
}
```

Select **all** calls to m that would **print out three or more values**.

- ☐ m(4, 7);
- ☐ m(4, 2);
- ☐ m(-1, 5);
- ☐ m(3, -1);
- ☐ m(1, 1);

Name: _____

(c) Consider the following code:

```
public static void mystery(int x, int y) {
    int z = 0;
    // Point A
    while (x > z) {
        // Point B
        z++;
        if (z % 2 == 0) {
            x = z * 2;
            // Point C
        } else {
            z++;
            // Point D
        }
    }
    // Point E
    System.out.println(x);
}
```

For each of the statements below, place a check (✓) in the corresponding box if it is true.

- ☐ At Point A, `x > 0` and `y > 0` must be true.
- ☐ At Point B, `x` must not be a negative number.
- ☐ At Point C, `z % 2 == 0` must still be true.
- ☐ At Point D, `x > z` must be true.
- ☐ At Point E, `x > 0` must be true.

Name: _____

2. Array Code Tracing

Consider the following method:

```
public static int[][] mystery(int[] list) {
    int[][] result = new int[list.length][list.length];
    for (int i = 0; i < result.length; i++) {
        int s = 1;
        for (int j = 0; j < result[0].length; j++) {
            result[i][j] = list[i] * (j+1) * s;
            s *= -1;
        }
        list[i] /= 2;
    }
    return result;
}
```

(a) Consider the following code:

```
int[] arr = {12, 13, 89};
int[][] result = mystery(arr);
```

What are the contents in `result` after this code is executed?

(b) Consider the following code:

```
int[] arr = {10, 5, 89};
int[][] result = mystery(arr);
```

What are the contents in `arr` after this code is executed?

(c) What statement is true about the `mystery` method? (Choose one)

- ☐ The `mystery` method fails during runtime if the passed parameter array `list` is empty, i.e. of size 0.
- ☐ If the passed parameter array `list` is empty, the resulting 2D array `result` returned will also be empty.
- ☐ All the elements (as in individual data type values) returned by the method in the 2D array `result` will have a positive (i.e. greater than zero) value.
- ☐ The number of elements (as in individual data type values) in the returned 2D array `result` will always be less than (or at least equal to) the number of elements in the passed parameter array `list`.

3. Debugging

Consider a static method called `complimentPet` that, based on pet information given by the user, forms and returns a String compliment for the pet.

The `complimentPet` method is passed the following as parameters:

- `Scanner scan` - The `Scanner` object to be used to gather further information about the pet. Using `scan`, the method prompts the user for the pet's age and species.
- `String petName` - The String name (all lowercase) of the pet that will be complimented!

Using these parameters, the method `complimentPet` forms a String compliment that is returned. Here are the specifications on how the compliment is decided:

1. The user is prompted for the pet's age in whole years and its species. Users input the pet's age using whole numbers, and input `1` if the pet is a cat or input `2` if the pet is a dog.
2. All compliment Strings start with the pet's name and `"is a"`. For example, if the pet's name is `"gigi"`, the compliment String will start with `"gigi is a"`.
3. Next, we'll use the pet's name to decide how to describe them:
   - If the pet's name has a length that is evenly divisible by 3, the pet is further described as `"adorable"`. For example, a pet named `"Skyler"` (whose name length is evenly divisible by 3) would have compliment String so far of `"Skyler is an adorable"`.
   - However, if the pet's name length is not divisible by 3 but instead has 1 character left over, the pet is described as `"lovely"`. For example, `"gigi"`'s compliment String so far would be: `"gigi is a lovely"`).
   - Similarly if the pet's name length is not divisible by 3 but instead has 2 character's left over, the pet is described as `"perfect little"`. For example, a pet named `"roxie"` would have a compliment String so far of `"roxie is a perfect little"`.
4. Now, given the pet's age, the compliment String is added to even further:
   - If the pet is younger than 7 years old and their name starts & ends with the **same** letter, they are described as a `"widdle baby"`. For example, a pet named `"bob"` that is 3 years old would be described as a `"widdle baby"`. Otherwise, the pet is only described as a `"baby"`.
   - If the pet is at least 7 years old, it is described as `"wise"` instead.
5. Finally, based on whether the user input `1` or `2` for the pet's species, the compliment is finished off with either `"kitty!"` or `"puppy!"`.

Here are a couple calls to `complimentPet` and their output (User input is in bold and underlined):

Suppose `complimentPet(scan, "gigi")` is called; this would be the user interaction:

```
What is gigi's age in years?:  7
Is gigi a cat or a dog?  Input 1 for Cat or 2 for Dog:  2
```

The method would return the compliment String, `"gigi is a lovely wise puppy!"`.

Suppose `complimentPet(scan, "bob")` is called; this would be the user interaction:

```
What is bob's age in years?:  5
Is bob a cat or a dog?  Input 1 for Cat or 2 for Dog:  2
```

The method would return the compliment String, `"bob is an adorable widdle baby puppy!"`.

Name: _____

Consider the following proposed buggy implementation of `complimentPet`. This implementation contains three bugs that are causing it to not work as intended!

Your task: Annotate (write on) the code below to indicate how you would fix the three bugs. You may add (using arrows to indicate where to insert), remove (by crossing out), or modify (with a combination) any code you choose. However, the fix should not require a lot of work.

- You must correctly identify three of the lines with issues, or correctly identify and fix two of the bugs for an S grade.
- You must correctly identify all three lines with the bugs and correctly fix all three of the bugs for an E grade.

```
1  public static String complimentPet(Scanner scan, String petName) {
2    System.out.print("What is " + petName + "'s age in years?: ");
3    int petAge = scan.next();
4    System.out.print("Is " + petName + " a cat or a dog? Input 1
       for Cat or 2 for Dog: ");
5    int petSpecies = scan.nextInt();
6    String compliment = petName + " is a";
7
8    if(petName.length() % 3 == 0) {
9      compliment += "n adorable ";
10   }
11   if (petName.length() % 3 == 1) {
12     compliment += " lovely ";
13   } else {
14     compliment += " perfect little ";
15   }
16
17   if(petAge < 7) {
18     char firstLetter = petName.charAt(0);
19     char lastLetter = petName.charAt(petName.length());
20     if(firstLetter == lastLetter) {
21       compliment += "widdle baby ";
22     } else {
23       compliment += "baby ";
24     }
25   } else {
26     compliment += "wise ";
27   }
28
29   if(petSpecies == 1) {
30     compliment += "kitty!";
31   } else {
32     compliment += "puppy!";
33   }
34   return compliment;
35 }
```

Name: _____

4. General Programming 1

Write a static method called `countPunct` that takes two parameters: a `Scanner input` and an integer `targetCount`, and returns an integer array for the count of punctuation marks present among the words. The method should prompt the user for words using the given `Scanner` until a certain total number of punctuation marks have been read and counted within the words that were input. The `countPunct` static method should return an array containing the count for each punctuation mark.

For this method, the five punctuation marks being searched for are: `?`, `!`, `:`, `[`, and `]`.

As such, the `countPunct` static method should return an integer array of size 5, with the counts of the punctuation marks in the order given above. After each word, if the `targetCount` number of total punctuation marks still hasn't been reached, a prompt for another word will be given.

For example, assuming the following declaration is made, the table below shows some sample calls to `countPunct` and resulting output (user input is **bold and underlined**):

```
Scanner console = new Scanner(System.in);
```

Call	`countPunct(console, 7);`	`countPunct(console, 1);`	`countPunct(console, 2);`
Output	`Give me word:` **Wow!** `Give me word:` **Gigi!** `Give me word:` **Why** `Give me word:` **are** `Give me word:` **you** `Give me word:` **[[such]]** `Give me word:` **a** `Give me word:` **good** `Give me word:` **girl??**	`Give me a word:` **Good** `Give me a word:` **luck** `Give me a word:` **on** `Give me a word:` **the** `Give me a word:` **exam;** `Give me a word:` **you** `Give me a word:` **can** `Give me a word:` **do** `Give me a word:` **it!**	`Give me a word:` **Have** `Give me a word:` **a** `Give me a word:` **restful** `Give me a word:` **spring** `Give me a word:` **break** `Give me a word:` **everyone!** `Give me a word:` **:)**
Return	`[2, 2, 0, 2, 2]`	`[0, 1, 0, 0, 0]`	`[0, 1, 1, 0, 0]`

You may assume that `targetCount` is always greater than 0, and that the user enters a single word with at least one character each time they are prompted.

You can also assume that input words are made up of alphanumeric characters (i.e. letters and numbers) and different punctuation marks (some being counted, some not). You must exactly reproduce the format of these logs.

Also, you don't have to make an `indexOf`-like method for this exam! Write your solution in the box on the next page.

Name: _____

Write your solution to problem #4 here:

Name: _____

5. General Programming 2

After their Lottery Dreams (on Quiz 2) failed, Matt and Elba have one more Hail Mary plan to make money: betting on coin flips!

Write a static method named `whatTheFlip` that accepts a `Random` object and an integer `headsInARow` and simulates coin flips (using the `Random` object) until you flip `headsInARow` heads. Your method should then return the total number of coins flipped, including the last flip. It should also print out a summary of what happened.

There are some specific notes about how this method should work:

- your coin flip should be "unbiased"; in other words, there should be a 50% chance of getting heads and a 50% chance of getting tails on each flip.
- you **may not assume that `headsInARow` is positive**. If the user passes in a value that is less than 1, your method should not flip any coins. Instead, print out an "Invalid" message and return −1.

Assuming that the following variable has been initialized:

```
Random randy = new Random();
```

Here are some example calls to the method with their resulting console output and return value:

Call	Console Output	Returned
`whatTheFlip(randy, 2)`	`Flips: H, T, T, H, H`	5
`whatTheFlip(randy, 2)`	`Flips: H, H`	2
`whatTheFlip(randy, 3)`	`Flips: H, T, H, T, T, H, H, H`	8
`whatTheFlip(randy, 0)`	`Invalid headsInARow: 0`	−1
`whatTheFlip(randy, -4)`	`Invalid headsInARow: -4`	−1

You must exactly reproduce the format of the console output shown above (other than whitespace), though the actual output may differ due to randomness.

You are not permitted to create any additional data structures (e.g. arrays, `ArrayList`s) to solve this problem. However, new variables for primitive data types are allowed. In addition, you should not declare a new `Random` object within your method; you must use the one passed in by the user.

Name: _____

Write your solution to problem #5 here:

Name: _____

6. Array Programming

Write a static method named `meanTwoArrays` that accepts two arrays of integers as parameters, calculates the combined average (mean) of all the elements in both arrays, and modifies each element in both arrays by subtracting that mean value. This method should not return anything. For example, consider the following:

```
int[] a1 = {8, 5, 10};
int[] a2 = {3, 5, 7, 4};
```

Calling `meanTwoArrays(a1, a2)` would return nothing; however, printing `a1` and `a2` would yield:

```
a1: [2, -1, 4]
a2: [-3, -1, 1, -2]
```

To break this example down, the sum of values of `a1` and `a2` is $8 + 5 + 10 + 3 + 5 + 7 + 4 = 42$. There are 7 total elements between the two arrays, so the mean is $42/7 = 6$. We obtain the result by subtracting 6 from each element in `a1` and `a2` (modifying the original arrays).

When calculating the mean value, truncate the fractional portion after dividing. In other words, you should use integer division when calculating the mean.

As shown in the previous example, the two input arrays may have different lengths. You will also need to handle cases where one of the arrays is empty (or both). For example, consider the following:

```
int[] a1 = {1, 2, 3, 4, 5, 6};
int[] a2 = {};
```

then the call of `meanTwoArrays(a1, a2)` would modify `a1` and `a2` like so:

```
a1: [-2, -1, 0, 1, 2, 3]
a2: []
```

Note that this example showcases how to truncate the mean. The sum of the values is $1 + 2 + 3 + 4 + 5 + 6 = 21$; using integer division the mean is $21/6 = 3$, which is why we subtract 3 from each item.

You are not permitted to create any additional data structures (e.g. arrays, `ArrayList`s) to solve this problem. However, new variables for primitive data types are allowed.

Name: _____

Write your solution to problem #6 here:

Name: _____

Name: _____

Just for fun: As a thanks for all their work, draw your TA a picture of what you think they will be up to during Spring Break!

TA Name: _____

Name: _____

This page is intentionally left blank and will not be graded; do not put exam answers here.