CSE 121 Spring 2024 Final Exam June 5, 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:**

# Q6: Array Programming

Write a static method called `insertMiddle` that takes in two integer arrays as parameters (we'll call them `arr1` and `arr2`) and returns a new array that contains:

1. the elements from the first half of `arr1`
2. then, all of the elements of `arr2`
3. finally, the rest of the elements in the second half of `arr1`

For example, consider the following two arrays:

```
int[] arr1 = {1, 2, 5, 6};
int[] arr2 = {3, 4};
```

A call to `insertMiddle(arr1, arr2)` should return the following array:

```
{1, 2, 3, 4, 5, 6}
```

It may be the case that the first array has an odd length. In this case, treat the first half as the "shorter half" of the two. For example, consider the following two arrays:

```
int[] arr3 = {2, 4, 6, 8, 10};
int[] arr4 = {1, 1, 1};
```

A call to `insertMiddle(arr3, arr4)` should return the following array:

```
{2, 4, 1, 1, 1, 6, 8, 10}
```

In addition,

- you may assume that neither array is `null`
- you may not assume that neither array is non-empty; in other words, your method should also work if one (or both) of the arrays has length `0`

As a reminder, you are restricted to the methods and classes provided on the reference sheet. Your method should not modify either of the arrays that are provided as parameters.

*Write your solution to problem #6 here:*

# CSE 121 Final Exam Reference Sheet

(DO NOT WRITE ANY WORK YOU WANTED GRADED ON THIS REFERENCE SHEET. IT WILL NOT BE GRADED)

```
type[] name = new type[length];
type[] name = {VAL1, VAL2, VAL3, ...};

type[][] name = new type[numRows][numColumns];
type[][] name = {
            {VAL1, VAL2, VAL3, ...},
            ...
            {VAL4, VAL5, VAL6, ...}
};
```

| Using Arrays | Description |
|---|---|
| `int value = name[i]` | Get the value at index `i` |
| `name[i] = value` | Set the value at index `i` |
| `name.length` | Get the number of elements in `name` |

| Using 2D Arrays (rectangular) | Description |
|---|---|
| `int value = name[i][j]` | Get the value at row `i`, column `j` |
| `name[i][j] = value` | Set the value at row `i`, column `j` |
| `name.length` | Number of rows (number of inner arrays) |
| `name[0].length` | Number of columns (length of inner array) |

| String Method | Description |
|---|---|
| `charAt(i)` | Returns character in this String at index `i` |
| `contains(str)` | Returns `true` if this String contains `str` inside it, returns `false` otherwise |
| `startsWith(str)` | Returns `true` if this String starts with `str`, returns `false` otherwise |
| `endsWith(str)` | Returns `true` if this String ends with `str`, returns `false` otherwise |
| `equals(str)` | Returns `true` if this String is the same as `str`, returns `false` otherwise |
| `equalsIgnoreCase(str)` | Returns `true` if this String is the same as `str` ignoring capitalization, returns `false` otherwise |
| `indexOf(str)` | Returns the first index this String where `str` begins, returns `-1` if not found |
| `length()` | Returns the number of characters in this String |
| `replace(str, newStr)` | Returns a new String with all `str` in this String replaced with `newStr` |
| `substring(i)` | Returns characters in this String from index `i` (inclusive) to end (exclusive) |
| `substring(i, j)` | Returns characters in this String from index `i` (inclusive) to `j` (exclusive) |
| `toLowerCase()` | Returns an all-lowercase version of this String |
| `toUpperCase()` | Returns an all-uppercase version of this String |

| Random Method | Description |
|---|---|
| `nextInt(max)` | Returns a random integer from 0 (inclusive) to `max` (exclusive) |
| `nextInt(min, max)` | Returns a random integer from `min` (inclusive) to `max` (exclusive) |

| Math Method | Description |
|---|---|
| `Math.abs(val)` | Returns the absolute value of `val` |
| `Math.max(val1, val2)` | Returns the larger of the two values `val1` and `val2` |
| `Math.min(val1, val2)` | Returns the smaller of the two values `val1` and `val2` |
| `Math.pow(base, exp)` | Returns the value of `base` raised to the `exp` power |
| `Math.sqrt(val)` | Returns the square root of `val` |

| Scanner Method | Description |
|---|---|
| `next()` | Returns the next token as a String |
| `nextLine()` | Returns the entire line as a String |
| `nextInt()` | Returns the next token as an `int`, throws exception if cannot |
| `nextDouble()` | Returns the next token as a `double`, throws exception if cannot |