| Object-Oriented Programming and Data Structures | Prelim 1 |
|---|---|
| CS 2110, Fall 2025 | Practice Exam |

# Version A

This exam consists of 9 problems (some with subparts) on a total of 14 pages, worth a total of 100 points. Please write all answers in the space provided below each question. You may also use the back of this page or the last page for additional space, but please mark on the question page itself if you are using this as part of an answer. **You have 90 minutes to complete this exam.**

This exam is closed-book and closed-notes. **Read and follow the directions to each question carefully** to be sure that you present your answers in the correct format and provide sufficient justification. For multiple choice questions, **fill in** the entire circle ◯ to indicate your selected answer.

Before you begin the exam, print your name and Net-ID on this cover page. Also, write your Net-ID in the space provided at the top of each page [1 point]. Finally, read and sign the academic integrity statement at the bottom of the page.

NAME: ______________________________ Net-ID: ____________

ACADEMIC INTEGRITY STATEMENT:

- All of the work I have written in the exam booklet is my own.
- I did not consult with any students or outside materials while taking the exam.
- I understand that academic integrity violations on this exam result in an F in the course.

SIGNATURE: ______________________________

**Net-ID:** ____________

This page can be used as extra space. Please mark on the question page itself if you are using this page for part of an answer.

**Net-ID:** ____________

**Problem 1. Static Types** [6 points]

Consider the following snippet of Java code:

```
int a = 2;
int b = 1;
double c = -6;

double d = b * b - 4 * a * c;
int n = -b + (b > 0 ? 1 : -1) * (int)Math.sqrt(d);
```

List the **static types** and **values** of each of the following expressions:

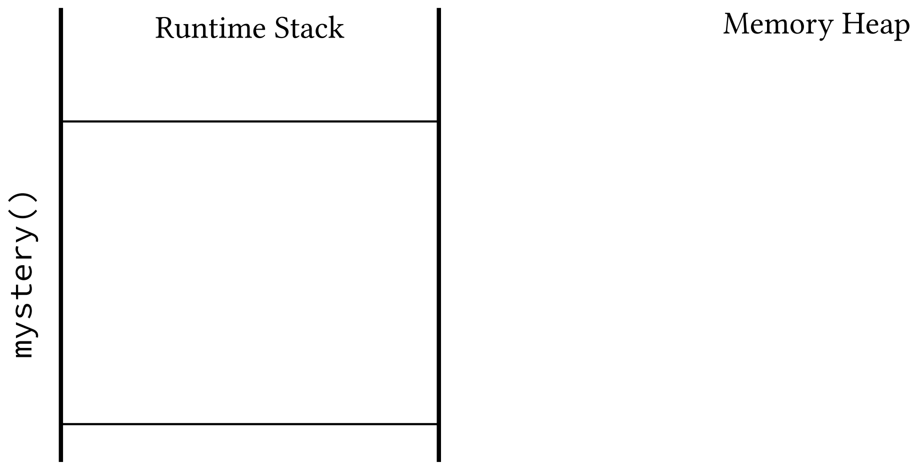
| Expression | Static Type | Value |
|---|---|---|
| `b * b - 4 * a * c` | | |
| `b > 0 ? 1 :-1` | | |
| `n / (2 * a)` | | |

**Problem 2. Memory Diagramming** [8 points]

Suppose that we execute the call `mystery(new String[]{"ghost", "bat", "wolf"})`, where `mystery()` is defined as follows (with specifications omitted):

```
public static String[] mystery(String[] words) {
    String[] ret = words;
    int mid = ret.length / 2;
    ret[mid] = words[0];
    return ret;
}
```

Use the space below to draw the state of the `mystery()` call frame *just before* the return statement is executed. Your diagram should include all heap objects reachable via references from this call frame (and no other objects).



**Net-ID:** ___________

**Problem 3. Multiple Choice** [$4 \times 2 = 8$ points]

(a) Which of the following is *not* an appropriate software engineering practice?

- ◯ Using `assert` statements to check for pre-condition violations.
- ◯ Writing unit tests before a class has been fully implemented.
- ◯ Writing unit tests that cause a checked exception to be thrown.
- ◯ Writing unit tests that cover pre-condition violations.

(b) Suppose that an exception propagates to the body of a `try`-block. The body of the first `catch`-block with which of the following properties will be executed?

- ◯ Its declared exception type is a *subtype* of the *dynamic* type of the thrown exception.
- ◯ Its declared exception type is a *subtype* of the *static* type of the thrown exception.
- ◯ Its declared exception type is a *supertype* of the *dynamic* type of the thrown exception.
- ◯ Its declared exception type is a *supertype* of the *static* type of the thrown exception.

(c) What is the *tightest* worst-case asymptotic upper bound on the *total number of recursive calls* made by the Merge Sort algorithm on an array of length $N$?

- ◯ $O(\log N)$
- ◯ $O(N)$
- ◯ $O(N \log N)$
- ◯ $O(N^2)$

(d) What is the *tightest* worst-case asymptotic upper bound on the *recursive depth* of the Merge Sort algorithm on an array of length $N$?

- ◯ $O(\log N)$
- ◯ $O(N)$
- ◯ $O(N \log N)$
- ◯ $O(N^2)$

**Problem 4. Recursion** [$10 + 3 = 13$ points]

In this problem, you'll use recursion to complete the definition of the following method:

```java
/** Returns the sum of all elements in `a` that are < 7. */
public static int sumBelow7(int[] a) { ... }
```

(a) To write a recursive definition without expensive array copying, you'll need to delegate to a recursive helper method with additional parameters. Write this method in the following box. Your answer should include

- ☐ The full signature (including visibility) of your helper method
- ☐ Complete JavaDoc specifications documenting its parameters and return value
- ☐ The completed body of your helper method

Your helper method must be *recursive* and *not iterative*; it may *not* include any loops, and it may *not* allocate any new arrays.

(b) Complete the body of `sumBelow7()` so it conforms to its specifications. Your answer should be a single line of code that returns the value of a single call to your recursive helper method.

```java
/** Returns the sum of all elements in `a` that are < 7. */
public static int sumBelow7(int[] a) {

}
```

**Problem 5. Inheritance and Type Hierarchies** [6 points]

Consider the following code (with specifications omitted):

```java
abstract class Equipment {
  public abstract void use();
}
abstract class ProtectiveGear extends Equipment {...}
abstract class Tool extends Equipment {...}
class PowerTool extends Tool {
  public void powerOn() {
    System.out.println("beep");
  }
  public void use() {
    System.out.println("bzzzzzzz");
  }
}
abstract class ManualTool extends Tool {...}
class Stapler extends ManualTool {
  public void use() {
    System.out.println("kachunk");
  }
}
class Hammer extends ManualTool {
  public void use() {
    System.out.println("bang bang");
  }
}
```

Draw the type hierarchy that shows the name of each class declared above and uses arrows to indicate inheritance. Arrows representing extending an abstract class should be dotted. Arrows representing extending a concrete class should be solid.

**Net-ID:** ____________

**Problem 6. Testing** [12 points]

We can separate characters into two categories:

- **Alphabetic** characters (a, é, Z, $\pi$, $\Theta$, etc.) comprise words in one or more languages.
- All other characters (8, #, –, 😁, whitespace characters, etc.) are **non-alphabetic**.

Suppose we define the following method to count the alphabetic characters in a `String`.

```
/** Returns the number of alphabetic characters in the given String. */
public static int countLetters(String s) { ... }
```

For example, `countLetters("a8Z–πa#")` should return 4.

Fill out the following table to describe 5 test cases for `countLetters()` (without using characters from the above definitions of alphabetic and non-alphabetic). For each test, describe its

- Input: The argument to `countLetters()`
- Output: The expected return value from `countLetters()`
- Explanation: A brief explanation of why you included this test.

Your explanations, taken all together, should persuade the reader that your test suite covers a variety of different kinds of possible inputs and outputs.

| Input | Output | Explanation |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |

**Problem 7. Subtypes** [$5 \cdot 3 = 15$ points]

Suppose we define the following interface to represent a collection of points on the real number line and two classes that implement this interface. Assume these classes compile successfully.

```
public interface PointSet {
    PointSet intersectWith(PointSet s);
    boolean enclosedBy(Interval i);
}

public class Interval implements PointSet {
    public Interval(double begin, double end) { ... }
    public double width() { ... }
    // ... additional field and method definitions ...
}

public class FinitePointSet implements PointSet {
    public FinitePointSet() { ... }
    public boolean addPoint(double p) { ... }
    // ... additional field and method definitions ...
}
```

Separately, we write client code that initializes the following four variables.

```
    Interval i = new Interval(0.5, 3.4);
    FinitePointSet f = new FinitePointSet();
    PointSet p1 = new Interval(-1.7, 2.8);
    PointSet p2 = f;
```

For each of the following expressions, determine whether it compiles and evaluates successfully at runtime, fails to finish evaluating due to a runtime error, or encounters a compiler error. Consider each statement independently.

- If the expression evaluates successfully, write its *static type* in the box.
- If the code has an error, identify the cause of this error in the box.

(a) `p2.addPoint(2.5)`

○ Compiles/Evaluates Successfully ○ Runtime Error ○ Compiler Error

Static Type / Error Cause:

(b) `i.intersectWith(p1)`

◯ Compiles/Evaluates Successfully ◯ Runtime Error ◯ Compiler Error

Static Type / Error Cause:

(c) `p1.enclosedBy((Interval) p1)`

◯ Compiles/Evaluates Successfully ◯ Runtime Error ◯ Compiler Error

Static Type / Error Cause:

(d) `p1.enclosedBy((Interval) p2)`

◯ Compiles/Evaluates Successfully ◯ Runtime Error ◯ Compiler Error

Static Type / Error Cause:

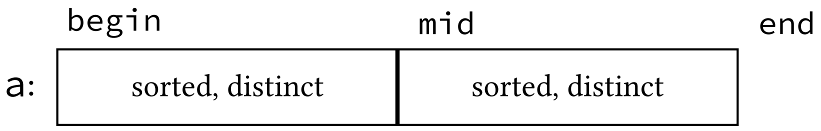
(e) `p1.enclosedBy((Interval) f)`

◯ Compiles/Evaluates Successfully ◯ Runtime Error ◯ Compiler Error

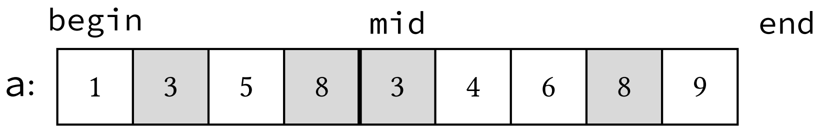
Static Type / Error Cause:

**Problem 8. Loop Invariants +** [$8 + 4 + 3 + 3 = 18$ points]

In this problem, we'll consider an array `a` with three indices `0 <= begin <= mid <= end <= a.length` such that the ranges `a[begin..mid)` and `a[mid..end)` are both sorted and neither of these ranges contains duplicate elements.



There may be some elements common to these ranges, which we call "matches." For example, there are 2 matches (3 and 8) highlighted in the following example:



(a) Complete the definition of the `firstMatch()` method according to its specifications. Your method should initialize loop variables `i` and `j` and complete the definition of the `while`-loop to maintain the following invariant:

*/* Loop inv: `a[begin..i)` and `a[mid..j)` have no common elements. */*

Your definition should not include a nested loop.

```
/** Returns the smallest index `i` with `begin <= i < mid` such
 *  that `a[i] == a[j]` for some `mid <= j < end`. Returns `mid`
 *  if `a[begin..mid)` and `a[mid..end)` have no matches. */
static int firstMatch(int[] a, int begin, int mid, int end) {
    int i =         ;
    int j =         ;

    while (                                        ) {




    }

}
```

(b) Fill in the loop body in the `countMatches()` method so that it conforms to its specifications. Your answer should include a call to the `firstMatch()` method, which you may assume works correctly regardless of your answer to part (a).

```
/** Returns the number of matches between `a[begin..mid)`
  * and `a[mid..end)`. */
static int countMatches(int[] a, int begin, int mid, int end) {
    int count = 0;
    int i = begin;
    while(i < mid) {







    }
    return count;
}
```

(c) Describe the invariant of your loop in part (b). You can do this by drawing an "Inv" array diagram or using words. In either case, your invariant should refer to all local variables of `countMatches()`.

(d) Suppose that $M =$ `mid - begin` and $N =$ `end - mid`. What is the *tightest* worst-case runtime complexity of `countMatches()` expressed in terms of $M$ and $N$? (No justification is required.)

**Net-ID:** ____________

**Problem 9. Class Design** [13 points]

Design a `Transcript` class to model a student transcript with the following requirements:

- `Transcript` objects should store the name of each course taken by the student and the grade received. We recommend using the `CourseGrade` record class on the next page.
- Client code should be able to add a new course and grade to the transcript.
- Valid course grades and overall GPAs are in the range [0.0, 4.3]
- Overall GPA is a simple (unweighted) average of all the course grades on the transcript.
- Client code needs to be able to access the overall GPA in O(1) time.
- Client code should be able to get the transcript as a formatted string suitable for printing an official transcript.

There should be no additional client-facing functionality outside what is described above.

A partial implementation, including all the methods needed for `Transcript`, is provided on the following page. Your job is to:

☐ Choose which fields the `Transcript` class will use to store its state, and add their declarations to the code.

☐ Write in JavaDoc documentation of class invariant(s) to ensure that `Transcript` objects are well-formed and self-consistent.

☐ Fill in the blank in front of each method with the correct visibility modifier for that method, and explain your answers here.

Rationale for visibility modifiers:

☐ Write the pre-condition for the `addGrade()` method, expressed as a condition that could be copied into a defensive programming assertion.

Complete these items by filling in the boxes within the provided skeleton code on the next page. Do not implement the method bodies.

**Net-ID:** ___________

```
/** Represents the `grade` (e.g., 3.7) that a student received in a
 *  course with the given `courseID` (e.g., "CS2110"). */
record CourseGrade(String courseID, double grade) { }

public class Transcript {

  // add field(s) and document class invariant(s) here










  /** Returns the letter grade (e.g., "B+") corresponding to the
   *  given `gpa` (e.g., 3.3). */
  [          ] static String gpaToLetter(double gpa) { ... }

  /** Returns the overall GPA, calculated as the unweighted average
   *  of all course grades on this transcript. Runs in O(1) time. */
  [          ] double overallGpa() { ... }

  /** Return a nicely formatted String with the course name and
   *  letter grade for each course in the transcript. */
   @Override
  [          ] String toString() { ... }

  /** Adds the course with the given `courseID` to this transcript
   *  associated with the given `grade`.
   *  Precondition for `grade`: Requires [                  ] */
  [          ] void addGrade(String courseID, double grade) { ... }
}
```

This page can be used as extra space. Please mark on the question page itself if you are using this page for part of an answer.