New York University
Department of Computer Science

CS101-004– Mock Midterm 2
Intro to Computer Science

Instructor: Tobias Blickhan

-

Name: ____
Net ID: ____

THIS IS A MOCK EXAM. IT DOES NOT COUNT TOWARDS YOUR GRADE.

The following instructions are as you get them in the real exam:

- Fill in your full name and NetID on this cover page.
- This exam contains 6 pages (including this cover page) and 7 questions. When the exam starts, make sure you are not missing any sheets.
- Total of points is 55. You have 55 minutes to answer all questions. This corresponds to roughly one point per minute. Manage your time.
- There are two blank pages at the end of the exam that can be used as scrap paper.
- If you have any questions during the exam or need more paper, raise your hand and I will get to you.
- It needs to be clear what question your answers correspond to. If it gets messy or if you write something on the scrap paper that should get graded, clearly indicate it. If you do not want something graded, cross it out. In the case where you give two answers to a question, and one is incorrect, you get no points.
- If necessary, comment your code so that it clear what you want to do. You do not get points for pseudo-code or comments, but you do get points for correct code that follows them.
- This exam is closed books, closed notes, and no electronic devices are allowed.

1. In the game of chess, pieces are moving on an 8×8 board. A rook can threaten another piece if the two share a row (rank) or column (file).

   The *n queens puzzle* is the problem of placing n chess queens on an $n \times n$ chessboard so that no two queens threaten each other. Queens can do all the moves that a rook can do, and move diagonally as well.

   In this problem, we implement a board for the simplified task of placing n rooks on an $n \times n$ board so that no two rooks threaten each other. Assume that all your definitions are placed within the class

```
    public static class NRooks {
```

   (a) (2 points) Implement the method `createBoard` that creates a square board filled with **`false`**.
   Example usage: `boolean [][] board = createBoard(8);`

   (b) (2 points) Implement the method `set` that changes a single value in the board to indicate that there is a rook there.
   Example usage: `set(board, 3, 4);`

   (c) (5 points) Implement the method `isThreatened` that checks if theres is a rook at a given position that threatens another rook.
   Example usage: `boolean threatened = isThreatened(board, 4, 1);`

   (d) (3 points) Implement the method `isComplete` checking if enough rooks have been placed to check for a valid solution.
   Example usage: `if (isComplete(board)){ ... }`

   (e) (3 points) Implement a method `isValid` that checks if no two rooks on the board see one another.
   Example usage: `boolean noRookThreatened = isValid(board);`

```
    } // end of class NRooks
```

2. (5 points) Implement a function that converts an ArrayList of strings to a two-dimensional array of characters. Here is a refresher on Lists and Strings:

```
public class ArrayList<T> {
    T at(int index);
    void add(T newValue);
    void remove(int index);
    int length();
}

public class String {
    char charAt(int index);
    int length();
    char[] toCharArray();
}
```

   The function signature is as follows:

```
public static char[][] toCharArrays(ArrayList<String> sentence) {
    // Your code here
}
```

3. Consider the following class definition:

```
public class Cat {
    ... // the code in here is not important for this problem
}
```

   (a) (2 points) Define classes `HouseCat` and `Tiger` that inherit from `Cat`.

   (b) (4 points) Write a method that takes an integer `N` and returns an array of `N` `Cat` objects. In particular, the elements are randomly either `HouseCats` or `Tigers` with equal probability. You can use this code snippet:

```
Random r = new Random();
boolean isHouseCat = r.nextBoolean();
```

   and you can assume that `import java.util.Random;` is at the top of your file.

   (c) (4 points) Write a method that takes an array of `Cat` objects and returns the number of `Tigers` in the array.

4. (a) (2 points) Implement a class `Book` that represents some basic details about a book in a bookstore: its title, author, price, and number of pages.

```
public class Book {
```

(b) (2 points) Implement a constructor for the class `Book` that takes the title, author, price, and number of pages as arguments.

(c) (1 point) Implement a constructor that does not require passing in the price, but allows for a default price of 9.99.

(d) (3 points) Implement getters and one setter method. Only the price of a book can be changed.

```
} // end of class Book
```

(e) (5 points) Create a new class `BorrowableBook` (which is also a Book!) that always has a price of 0.00. A `BorrowableBook` also stores the name of the person who borrowed it as well as the days remaining until it is due for return. Implement a constructor as well as getters and setters where necessary.

```
public class BorrowableBook
```

(f) (1 point) Add a method `isLate` that determines if a book is overdue for return.

```
} // end of class BorrowableBook
```

(g) (3 points) Now represent the Invetory of a bookstore. Implement a class `Inventory` that stores an ArrayList of books as well as the money in the register. Add a default constructor.

```
public class Inventory {}
```

(h) (1 point) Implement a method `addBook` that adds a book to the inventory.

(i) (1 point) Implement a method `buyBook` that buys a book at 80% of the book's price using money in the register. Do not worry about fractional cents. It is okay to go into debt.

(j) (6 points) Implement a method `sellBook`. This method takes in a title and removes the book from the inventory. The money from the sale is added to the register. If the book is borrowable, set the due date to 14 days. The borrower needs to also be set and therefore needs to be passed as a parameter to `sellBook`.

```
} // end of class Inventory
```

5. (0 points) (Extra practice) Write a method that, given an array of `Circle` objects, sorts the circles from smallest to largest. Assume that the `Circle` class is defined as follows

```java
public class Circle {
    private float radius;

    public Circle (float r) {
        if (r >=0) radius = r;
        else radius = 1;
    }

    public float getRadius() {
        return radius;
    }
    public int compareTo(Circle c) {
        if (radius == c.radius ) return 0;
        else if (radius < c.radius ) return -1;
        else return 1;
    }
}
```

The signature for your method should be `void sortCircles (Circle[] circles)`.

6. (0 points) (Extra practice) In the following code snippet, the variable `n` appears in a number of scopes.

```java
public class manyN {
    private int n = 10;
    public int f() {
        int n = 1;
        return n;
    }
```

Is this an admissible assignment? What is the value of `n` that is returned? How would one access the `n` from the class - the one with value 10?

```java
    public int g(int k) {
        int a;
        for (int n = 1; n <= k; n++)
            a = a + n;
        return a;
    }
```

Is this an admissible assignment?

```java
    public int h(int n) {
        int b;
        for (int n = 1; n <= 10; n++)
            b = b + n;
        return b;
    }
} // end of class manyN
```

Is this an admissible assignment?

7. (0 points) (Extra practice) What is the output of the following code snippet?

```java
int [][] array = { {1, 2,}, {3, 4}, {5, 6} } ;

System.out.println( array.length + ", " + array[0].length + "\n" );

for (int i = 0; i < array.length; i++){
    System.out.print ( array[i][0] + ", " );
}
```