CSCI-UA.0101-007: Practice Midterm Exam 1

Duration: 75 minutes

Instructor: Michael Tao
New York University

October 16, 2023

Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, use the extra pages at the end of the exam.

Name: _____

NetID: _____

Do not open this exam until instructed.

Instructions:

- **Write your full name and your NetID on the front of this exam.**
- Make sure that your exam is not missing any sheets. There should be (11) double sided pages in the exam.
- Write your answers in the space provided below each problem. If you make a mess, clearly indicate your final answer (or state that it is on the scrap paper).
- If you have any questions during the exam, raise your hand and we will get to you.
- At the end of the exam, there is one blank page. Use this as your scrap paper. If you need additional scrap paper, raise your hand and we will get it for you.
- This exam is closed books, closed notes, and no electronic devices are allowed.

Good luck!

```java
// Some useful java functions

// withih java.util
public class Arrays {
    // sorts the elements of arr
    public static int[] copyOf(int[] arr) {}
}

// withih java.util
public class Scanner {
    String nextLine() {  }
    void close() {  }
}

public class ArrayList<String> {
    int size() {}
    void add(String element) {}
    // This is not technically part of ArrayList but lets ignore that part...
    String[] toArray() {}
}

public class Random {
    // returns a value between 0 and bound (including 0 but not including bound)
    int nextInt(int bound);
}

public class String {
    String[] split(String regex) {  }
    boolean equals(String other) {  }
    int length() {  }
    char charAt(int index) {  }
    String substring(int start, int end) {  }
    String trim() {  } // removes whitespace before and after
}

// example usage of an array and printing out its length
int[] x = {0,1,2};
int[] y = new int[3];
int length = x.length;
System.out.print("X␣length␣");
System.out.println("=␣" + length);
// prints "X length = 3"



// common Unix commands:
ls
mv
cp
cd
pwd
touch
```

Problem 1: ls output

(1) What do . and .. mean in the output of ls −a?

(2) What are the permissions for mtao and cims in

```
−rwxrwxr−x 1 mtao cims 21160 Oct 13 15:52  main.tex
```

Problem 2: Patiently Waiting

Implement the body of a function that uses a while loop to continuously listen to lines typed by a user until they say the lines "yes" or "no". If they say "yes" return true and if they say "no" return false. The user must use the right capitalization but spaces are allowed

```
    Potential input:
hi
yes // <— function then returns true

Potential input:
NO
No
 no // <— function returns false
```

(1) Implement this function using if/else statements.

```
public static boolean userChoice() {
    Scanner scnr = new Scanner(System.in);









}
```

(2) Implement this function using a switch/case statement.

```
public static boolean userChoice() {
    Scanner scnr = new Scanner(System.in);









}
```

Problem 3: Non-Zero Entries

Implement the body of a function that takes in a list of integer values and returns the number of them that were non-zero.

```
public static int nonZeroValues(int[] values) {




}
```

Problem 4: Sorting

(1) Implement a function that sorts an array the input data sorted, but does not modify its input. Don't forget to fill in the return types! You can use existing sorting functionality.

```
import java.util.Arrays;

// int[] data = {5,4,2,3,1};
// int[] sorted = sort(values); // sorted should be {1,2,3,4,5}
public static _____ sort(int[] values) {




}
```

(2) Implement a function that sorts an array the input data sorted by modifying its input. Don't forget to fill in the return types! You can use existing sorting functionality.

```
import java.util.Arrays;

// int[] data = {5,4,2,3,1};
// sort(values); // values should be {1,2,3,4,5}
public static _____ sort(int[] values) {




}
```

Problem 5: Null-Terminated Strings

In languages like C, the special character '\0' called "nil" indicates the end of a string, so even if there is more data after a '\0' character it is ignored. For instance "Hello\0World" is interpreted as "Hello". Java does not use these "null terminated" strings. Write a function that converts a null-terminated string to Java-style string by only returning the data before the first '\0'.

```
// strip("Hello!\0") == "Hello!"
// strip("\0") == ""
// strip("\0jt32iojgq9gq904hjq") == ""
// strip("John\0Doe") == "John"
public static String strip(String input) {









}
```

Problem 6: Checkerboard

Implement the body of a function that prints out a checkerboard comprised of the characters x and o according to a prescribed number of rows and cols.

Invoking checkerboard(3,5) should print out

x0x0x
0x0x0
x0x0x

```
public static void checkerboard(int rows, int cols) {









}
```

Problem 7: Cantor's diagonal argument

Cantor's diagonal argument has been used to prove a wide variety of important theorems like Godel's incompleteness theorem (the existence of unprovable statements in any logic system) and the unsolvability of the Halting problem (the ability to predict whether a program will ever finish on a given input). In this problem we will use his argument to quickly make a word that does not exist in a list of words of the same length and assume that we have the same number of words as characters in each word.

The trick to his diagonal argument is that we can construct a new word by taking the i^{th} character of the i^{th} word. For example, if we have the words "catch","bikes","south", "mouse","soupy" we can construct the word by NOT taking 'c' from "catch", 'i' from "bikes", etc. to obtain "djvtz".

The input should be an array of 'String' objects and the output should be a single string that does not exist in the sequence.

```
public static _____ getNewWord(_____) {

}
```

Problem 8: Bounding Boxes

Write a function that, given three arrays holding pairs of numbers p, min, max, detects if both of the points in p lies in the box defined by min and max. A point lies between min and max if $p[0]$ lies between $min[0]$ and $max[0]$ and $p[1]$ lies between $min[0]$ and $max[1]$. If a point lies inside then return true, otherwise return false.

```
public static _____ inBoundingBox(_____) {
```

}

Problem 9: Manhattan Distance to a Box

For two points (x, y) and (z, w) the Manhattan distance (also called L^1 distance or taxicab distance) is defined as $|x-z|+|y-w|$. It measures the distance a car (or taxicab) must take to travel between two places in a city whose roads lie on a grid (like Manhattan).

Let us define the distance to a rectangular neighborhood as following: if we are inside the neighborhood our distance is 0. Otherwise we compute the Manhattan distance to the nearest point on the rectangle, which is on the boundary. Similar to the last problem we write the rectangle (or box) as two arrays *min* and *max*.

Lets solve this problem in two steps: first by implementing one dimensional Manhattan distance to an interval (a 1D rectangle), and then using that code to implement the distance in two dimensions. One dimensional Manhattan distance between two points x and z is defined as $|x - z|$.

```
public static _____ distanceOneDimension(_____) {









}
public static _____ distanceTwoDimension(_____) {









}
```

Scrap paper **Name**: _____

Scrap paper **Name**: _____

Scrap paper **Name**: _____