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 (14) 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 void sort(int[] arr) { ... }
}

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

public class ArrayList<String> {
    int size() {}
    void add(String element) {}
    String get(int index) {}
    // 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 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) Busses connect different computer components like the CPU and Memory together. **(T / F)**

(2) ASCII is the only encoding used in text files. **(T / F)**

(3) When you run ls it shows the files in the current working directory. **(T / F)**

(4) 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: 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 8: 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(_____) {




}
```

Problem 9: Moving unix files

(1) Say we want to organize some assignments on a filesystem. Say we start with following files relative to our working directory:

```
./cs/101_old/a2.txt    ./cs/101/a1.txt    ./cs/101/a2_old.txt
./cs/101/a2.txt        ./cs/101/a3.txt
```

After you finish this sequence of questions your filesystem should only contain the following files:

```
./cs/101/a1.txt    ./cs/101/a2.txt    ./cs/102/a1.txt
./cs/101/a4.txt    ./cs/102/a1.txt
```

Answer each question assuming you did not change your working directory.

(1) Provide commands to delete the old files and directories (all entries with _old in their names).

(2) Make an empty new file a4.txt in the 101 folder.

(3) Make an empty new file a1.txt in a new directory 102 that lies in the cs folder.

(4) Provide commands to move the third assignment for cs101 to be the first assignment in cs102. In other words, ./cs/101/a3.txt should be placed at ./cs/102/a1.txt

Problem 10: Clamp

The clamp function restricts a number x to a prescribed range $[min, max]$. In particular, when x lies in the range then the clamp function returns x, when x is less than min then the function returns min, and when x is greater than max it returns max. It is assumed that $min < max$.

```
// Example results:
// 0.5 == clamp(0.5,0.0,1.0);
// 0.0 == clamp(-2.0,0.0,1.0);
// 1.0 == clamp(20.0,0.0,1.0);
public static double clamp(double x, double min, double max) {









}
```

Problem 11: Sycophant

Implement the body of a function that prints "yes" indefinitely in a while loop while keeping track of the number of times it's done so. The output must match the example output. The function should never stop running.

```
// Example output:
1) yes
2) yes
3) yes
...
100) yes
...
public static void sycophant() {









}
```

Problem 12: RGB

The Red-Green-Blue (RGB) scheme for representing colors is quite prevalent but it takes some time to get used to different colors additively. Usually each color "channel" is represented by a single byte, so the entire color can be represented by a byte[3]‘ array.

One common tool is to use aliases to hide the use of the scheme. Write a function that, upon receiving one of the aforementioned colors, returns a new array with the respective color. You must use a switch/case to handle the example results below. If the color is not recognize it just return the code for black.

```
// Example results:
// {255,0,0} == getRGB("red");
// {0,255,0} == getRGB("green");
// {0,0,255} == getRGB("blue");
// {0,0,0} == getRGB("black");
// {255,255,255} == getRGB("white");
public static byte[] getRGB(String name) {




}
```

Problem 13: Bogosort

The worst algorithm for sorting an array of numbers is called bogosort, which repeatedly randomly shuffles the input array of numbers until until it is sorted. To help with this we will implement two utility functions: isSorted and shuffle.

(1) Implement isSorted, which checks that the input array is sorted. Note that this is equivalent to checking that every pair of adjacent entries are sorted. That is, for any i, $value[i]$ is less than $value[i+1]$. Be mindful to only access entries within the array and not accidentally try to access data beyond its end.

```
// Sample input:
// int[] data = {0,3,-2};
// false == isSorted(data);
// int[] data2 = {0,3,6};
// true == isSorted(data);

public static ________ isSorted(_____________) {









}
```

(2) Implement shuffle, which creates a randomly moves values around. There are a large number of algorithms possible, but for this exam we will use a form of "random swapping". In particular, for every number in our array we will randomly pick another number in the array and swap their values. The new random value should be generated by calling random.nextInt. This function should modify the input data and not return anything.

```
public static ________ shuffle(_____________) {
    Random random = new Random();









}
```

(3) Finally, implement the bogosort function. It should return an array and NOT modify the input array. Recall that it calls shuffle until isSorted reports true. Then it will return the array.

```
// int[] data = {0,3,-2};
// int[] sorted = bogosort(data);
// data is still {0,3,-2}
// sorted == {-2,0,3}

public static ________ bogosort(_____________) {









}
```

Problem 14: Polynomial Evaluation (Bonus)

Naive evaluation of the polynomial $ax^3 + bx^2 + cx + d$ requires $3 + 2 + 1 = 6$ multiplications and 3 additions, as can seen in

$$a * x * x * x + b * x * x + c * x + d.$$

For a polynomial with k coefficients it is implemented by adding multiplying the i^{th} coefficient by x a total of $k - i - 1$ times and accumulating these sums.

However, we can rewrite this polynomial as

$$(((a * x) + b) * x + c) * x + d,$$

which takes only 3 multiplications and 3 additions with what's called Horner's method. Here we start with the first coefficient, and then repeatedly multiply by x and adding the next coefficient to the result.

$$\begin{aligned} r &= a \\ r &= r * x + b \\ r &= r * x + c \\ r &= r * x + d. \end{aligned}$$

At the end r is the result of evaluating the polynomial. Horner's method turns out to require less code to implement and is also more accurate.

Implement both versions of polynomial evaluation, without assuming the total number of coefficients without using any math utilities like Math.pow.

```
// double[] C = {4.0,3.0,2.0,1.0};
// double y = naive_polynomial(C, 0.5); // 4 * .125 + 3 * .25 + 2 * .5 + 1
// double[] C2 = {5.0,0.0};
// double y2 = naive_polynomial(C2,0.5); // 5 * .5

public static _____ naive_polynomial(_________________) {









}
```

```
// double[] C = {4.0,3.0,2.0,1.0};
// double y = horner_polynomial(C, 0.5); // ((4 * .5 + 3) * .5 + 2) * .5 + 1
// double[] C2 = {5.0,0.0};
// double y2 = horner_polynomial(C2,0.5); // 5 * .5

public static _____ horner_polynomial(_________________) {









}
```

Scrap paper **Name**: _____

Scrap paper **Name**: _____

Scrap paper **Name**: _____