Submit Exam 1 | Gradescope

Read the exam policies. You have until 12:15pm eastern time to finish this exam. Setting an alarm is recommended. Write your name below to acknowledge the exam policies (and for free points).

Q2 Short Answer Questions

12 Points

Q2.1 To Infinity and Beyond?

4 Points

In math, there is no largest or smallest integer; they go on infinitely. However in Java, there *is* a largest and smallest `int`: `Integer.MAX_VALUE` and `Integer.MIN_VALUE` respectively. Briefly explain why `int`s cannot represent all possible integers.

Q2.2 Binary Search Variant

4 Points

Consider this `binarySearch` function, a slight variant on the iterative version seen in class.

```
public static int binarySearch(int[] elements, int target) {
    int left = 0;
    int right = elements.length;
    while (left < right) {
        int middle = (left + right) / 2;
        if (target < elements[middle]) {
            right = middle;
        } else if (target > elements[middle]) {
            left = middle + 1;
        } else {
            return middle;
        }
    }
    return -1;
}
```

In the version in class, `left` and `right` were the first and last index of the portion of the array we were searching. What do `left` and `right` represent in this version?

Q2.3 For vs. While

4 Points

Anything you can do with a for loop, you can also do with a while loop. Briefly explain why.

Q3 Wack-a-Mole but it's PennDraw

18 Points

TA Adi has been acting up during staff meetings, so Becca decided to quickly code up a game in order to distract him. Unfortunately, some of her lines jumbled up and she needs your help to unscramble them to make sure Adi's short attention span doesn't keep interfering with her staff meetings.

The game is simple – a circle is drawn; once you click the circle, a new, smaller, circle is redrawn randomly in another part of the screen and the process repeats for each successive circle.

```
public class Game {
    public static void main(String[] args) {
        PennDraw.enableAnimation(30);
        PennDraw.setPenColor(PennDraw.BLUE);
        double radius = 0.5;
        double px = Math.random();
        double py = Math.random();
        while (true) {
          // scrambled lines go here
        }
    }
}
```

Unscramble the lines in the body of the `while` loop. Each of the following lines (in alphabetical order) are used once.

```
double mx = PennDraw.mouseX();
double my = PennDraw.mouseY();
if (PennDraw.mousePressed()) {
if (mx > px - radius && mx < px + radius && my > py - radius && my < py + radius) {
PennDraw.advance();
PennDraw.clear();
PennDraw.filledCircle(px, py, radius);
px = Math.random();
py = Math.random();
radius *= 0.75;
}
}
```

Write the completed main function here:

Q4 Buggy Phone Numbers

18 Points

Kishen has so many friends to keep in touch with over the summer, so luckily he has everyone's phone number. He keeps the phone numbers in one long array of characters, like how we all keep track of our friends' numbers.

BUT, Kishen accidentally turned the numbers into letters! Each index of the array has a letter 'a'-'j' which corresponds to a single digit number. ('a' is 0, 'b' is 1, 'j' is 9, and etc. for every letter in between)

There is a catch! The first digit of each phone number is correct, i.e. they are NOT letters and they are chars of numbers with the correct digit. To fix this array, the first digit of each 10-digit phone number does not need to be changed. For example, his broken array containing 2 10-digit phone numbers may look like `char[] brokenNumbers = {'1', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', '2', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'};`, where the original phone number array should look like `char[] fixedNumbers = {'1', '0', '1', '2', '3', '4', '5', '6', '7', '8', '2', '1', '2', '3', '4', '5', '6', '7', '8',`

```
'9'};.
```

Kishen wrote this code to turn the letters back into numbers so he could send his friends GamePigeon checkers, but he struggled a little so there are a bunch of bugs. Help Kishen find the bugs in his code!

```
1. public static void fixPhoneNumbersWrong(char[] nums) {
2.       fixPhoneNumbersHelper(nums, 1);
3.  }
4.  public static void fixPhoneNumbersHelperWrong(char[] nums, int index) {
5.      if (index == 0) {
6.          fixPhoneNumbersHelper(nums, index + 1);
7.      } else if (index > nums.length) {
8.          return;
9.      } else {
10.         nums[index] = (char)(nums[index] - ('a' - '0'));
11.     }
12. }
```

There are some bugs in this code. You need to find 3 of them. Be sure to identify the line number and a brief description of the bug.

Q5 Decoding the Ye Olde Phone Keypad

20 Points

After fixing the code in question 4 for Kishen, Botong realized her phone is also having issues. Her phone keyboard seems to be broken, and will only give her access to the keypad, so she's been sending messages using the letters written on the keypad:

For example, to type the letter A you would press 2 once, and to type the letter Z you would press 9 four times. Unfortunately, yet another phone bug resulted in Botong's messages being sent out as numbers, instead of letters.

You are given the following helper function which converts a single group of numbers representing one letter into that letter. You do not need to understand the implementation of this function, as it uses features we have not yet seen in the course. For example, `seqToChar("222")` returns `'C'`.

```
/* Description: converts a sequence of numbers into the character it
 * represents on the phone keypad.
 * Input: a string containing a valid sequence of numbers
 * Output: the corresponding letter in uppercase
 */
public static char seqToChar(String seq) {
    char[][] chars = {
                        {'A', 'B', 'C'},
                        {'D', 'E', 'F'},
                        {'G', 'H', 'I'},
                        {'J', 'K', 'L'},
                        {'M', 'N', 'O'},
                        {'P', 'Q', 'R', 'S'},
                        {'T', 'U', 'V'},
                        {'W', 'X', 'Y', 'Z'}};
    return chars[seq.charAt(0) - '2'][seq.length() - 1];
}
```

Your task is to write the function `seqToString` to convert a string of these numbers into the appropriate string representing the actual English message Botong meant to send, in uppercase. Each group of numbers

representing a letter is followed by a space, signaling that the letter is complete. Note that a space is just a regular `char`, denoted by `' '`. To check if the `i`th character of a string `s` is a space, you can use `if (s.charAt(i) == ' ')`.

You can assume that all inputs to this function will be valid keypad characters, e.g. there will be nothing like “0” or “2222” or “abc” inside the string given as input, and they are formatted correctly, with a space after each "letter".

For example, `seqToString("44 33 555 555 666 9 666 777 555 3 ")` should return the string `“HELLOWORLD”`.

```
/*
 * Description: converts a string of keypad numbers into a message
 *
 * Input: the string containing sequences of numbers, assumed to be valid
 * Output: the message that the input string represents in uppercase
 */
public static String seqToString(String input) {
    // your code here
}
```

Complete the body of the function and write it below:

Eli and Paul are having a heated debate about whether tabs or spaces are the superior way to indent code, a classic programming dilemma. They decide to settle their debate the old fashioned way: war (the card game). In the heat of the moment, they forgot to keep track of the score, but luckily, they have a record of each round in the form of an `int` array. The even indices in the array correspond to the cards that Eli has played, and the odd indices in the array correspond to the cards that Paul has played. The cards are numbered 1-13, with Ace corresponding to 1. Every two indices corresponds to a round played. The winner of each round is the player who draws the highest card. The king (13) is the highest card, and the ace (1) is the lowest. In the result of a tie in a round, no one wins that round.

For example, a two round game in which Eli wins the first round and Paul wins the second round can be written as `int[] twoRoundGame = {10, 4, 9, 12};` In the first round, Eli played 10 and Paul played 4, so Eli won. In the second round, Eli played 9 and Paul played Queen (12), so Paul won. If there is a tie in the entire game, Paul wins, because he's the instructor.

Write a function that checks whether the input array is valid or not. Return true if it’s valid and false if it is invalid. You should check if the array contains complete rounds (where Paul and Eli both drew a card). You should also check that every card is valid (between 1 and 13).

```
/*
 * Description: checks whether the input contains
 * complete and valid rounds.
 *
 * Input: the game input int array
 * Output: true if the array is valid, false otherwise
 */
public static boolean isValidArray(int[] input) {
    // your code here
}
```

Complete the body of the function and write it below:

Q6.2 Result Test Case

4 Points

In question 6.3, your job will be to write the function `resultOfWar` that takes in the `int[]` of results and returns the name of the player who won the game as a `String` of either `"Eli"` or `"Paul"`. Write a JUnit test case testing `resultOfWar` that corresponds to the `twoRoundGame` example above. Assume the `import` statements are given and that the class is written for you. i.e. your code snippet should start with `@Test`. Note that `assertEquals` is smart, and compares `String`s appropriately, not using `==`.

Q6.3 Result of War

14 Points

Now, your job is to write `resultOfWar`. You should call your function from part 1 to check whether the array is valid. If it isn't valid, return "Error: invalid input".

```
/*
 * Description: determines player who has won the most rounds
 * in the game
 *
 * Input: the game input int array
 * Output: name of the winner of the array
 */
public static String resultOfWar(int[] input) {
    // your code here
}
```

Complete the body of the function and write it below:

Q6.4 Random Rounds

14 Points

Paul and Eli got bored of drawing cards themselves and wanted to write a program to generate new rounds. Fill in the function so it returns a new array that has double the rounds. The first half of the new array will be identical to the input, and the second half should be randomly generated. Each number in the randomly generated array should be an `int` between 1-13, inclusive.

```
/*
 * Description: adds new rounds to the input array
 *
 * Input: the game input int array
 * Output: an array of twice the length, with the second half being
 * randomly generated new rounds
 */
public static int[] addNewRounds(int[] input) {
    // your code here
}
```

Complete the body of the function and write it below: