CIS 110 Midterm I Mar 5, 2020

SOLUTIONS

1. **Relaxation** (3 points)

   (a) Close your eyes

   (b) Breathe in deeply through your nose

   (c) Breathe out slowly through your mouth

   (d) Repeat until you are told to start the exam

2. **Variables, Types, and Iterations (10 points)**

(a) What is the value of the following expression `'x' - 'z'`? (2 points)

(a) ☐ `2`
(b) ☒ `-2`
(c) ☐ `-1`
(d) ☐ `'y'`
(e) ☐ This is not allowed in Java

(b) What is the only value of `x` that will make this expression `true`? (2 points)

```
x % 3 == 2 && x <= 9 && x > 2 && x / 6 == 1?
```

(a) ☐ `5`
(b) ☐ `2`
(c) ☐ `6`
(d) ☒ `8`
(e) ☐ No such x exists

(c) Which of the following is the proper way to create a `boolean array b` of length 6? (2 points)

(a) ☐ `boolean[6] b = new boolean[];`
(b) ☐ `boolean[] b = boolean[6];`
(c) ☒ `boolean[] b = new boolean[6];`
(d) ☐ `boolean[] = new boolean[6];`
(e) ☐ None of the above

(d) What is printed by the code below? (2 points)

```
int[] a = {9, 1, 7, 5, 3};
for (int i  = 0; i < a.length; i++) {
    if (a[i] > i) {
        continue;
    }
    System.out.print(a[i] + " ");
}
```

(a) ☐ `9 1 7 5 3`
(b) ☐ `1 7 3`
(c) ☐ `1 5 3`
(d) ☐ `9 7`
(e) ☒ None of the above

(e) Add the correct data type for the following variables. Pick only one if multiple data types apply. (2 points)

```
boolean x = 6 % 2 != 3 /2;

String x = 4 + "4";

double x = 4 % 3 + 7.2 -1;

boolean x = 4 % 3 + 7.2  < 1 * 0.5;
```

3. **Definitions, functions, recursion, String, and Iterations (22 points)**

(a) This is a case of valid method overloading.
Yes ☐ No ☒ (2 points)

```
public static int imran(int x, int y) {...}
public static double imran(int a, int b) {...}
```

(b) This code snippet below prints `The average on this exam is 88 out of 110.`
Yes ☐ No ☒ (2 points)

```
String first = "The average on this exam is ";
String second = 8 + 8 + " out of " + 1 + 1 + 0;
System.out.println(first + second);
```

(c) A recursive function with return type `double` must have a `return` statement in the body of its **base case**.
Yes ☒ No ☐ (2 points)

(d) A `boolean` variable can be implicitly cast (converted) into an `integer` variable.
Yes ☐ No ☒ (2 points)

(e) This is a case of valid method overloading.
Yes ☒ No ☐ (2 points)

```
public static int imran(int x, boolean y) {...}
public static int imran(boolean a, int b) {...}
```

(f) When **passing an array variable to a function**, a copy of the array is passed to the functions. Any changes made to the array inside the function are not visible outside of the function.
Yes ☐ No ☒ (2 points)

(g) A recursive function can only have one base case.
Yes ☐ No ☒ (2 points)

(h) Writing `int[] arr = 1, 2, 3;` is a valid way to initialize an array of integers of length 3.
Yes ☐ No ☒ (2 points)

(i) All for loops can be written using a while loop.

Yes ☒ No ☐ (2 points)

(j) The code below will print `i` 5 times.

Yes ☒ No ☐ (2 points)

```
for (int i = 10; i > 0; i /= 2) {
        System.out.println(i);
}
```

(k) Match each concept with the correct definition. Write the answer number next to the concept definition answers.(2 points)

- Declaration statement: _____(3)_____
- Variable: _____(2)_____
- Assignment statement: _____(4)_____
- Expression statement: ___(1)_____

(1) is a statement that evaluates to a value

(2) is a name that refers to a value

(3) associates a variable with a type

(4) associates a value with a variable

4. **Debugging (12 points)**

There are 6 errors in the code below including the command line call (in the interactions pane in DrJava). Write the line number and a brief description (in your own words) of what you think is causing the error.

```
public class BuggyAggregate{
/* aggregate takes in a double array and a minimum value.
 * It finds the sum of all values in the double array greater than the
 * minimum value.
 */
1.  public static int aggregate(double[] arr, min ){
2.    int i = 0;
3.    if(true){
4.      sum = 0;
5.      for(i; i < arr.length; i = ((i + 1 - 1) / 1) + 1){
6.        if(input < arr[i]) {
7.          sum += arr[i];
8.        }
9.      }
10.     return sum;
11.   }
12.  }

13.  public static void main(String[] args){
14.    int input = args[0];
15.    double[] arr = {0.0, 2.0, 3.0, 5.0, -10, 'a', 1.0 / 0.0 };
16.    String aggregated = aggregate(arr, input);
17.  }
}

//==== Interaction Pane ====
18. java BuggyAggregate 2
```

	Line number	Error (description)
(1)	12	Missing return statement
(2)	6	Variable out of scope/not declared
(3)	4/5	Variable not declared
(4)	16	Wrong assignment of return type
(5)	14	Data type mismatch
(6)	1	Missing variable type in function declaration

5. **Recursion (24 points)**

(a) **Max vs Recursion** (12 points)

Max wrote some code the other day, but he shook his computer too hard (Jules's idea) and many lines of code fell out of the DrJava file. He needs some help **filling in the blanks**.

Max's homework for CIS100 was to write a RECURSIVE FUNCTION that determines whether or not a String is a subsequence of another String.

For instance:

`"abcd"` is a subsequence of `"aebfcsd"`

`""` (empty string) is a subsequence of `""`

`""`(empty string) is a subsequence of `"abcde"`

`"abcde"` is a subsequence of `"abcde"`

`"abc"` is a subsequence of `"dabce"`

`"abc"` is NOT a subsequence of `"bca"` `"acbd"` is NOT a subsequence of `"aebfcsd"`

Consider the following module definition

```
public static boolean isSubSequence(String substring, String search,
                                    int subStringIndex, int searchIndex){
    // base case for when we exhaust the substring string

    if( subStringIndex >= substring.length() ){
        return true;
    }
    // base case for when we exhaust the search string

    if( searchIndex >= search.length() ){
        return false;
    }
    // recursive call: we found a match between substring
    // and the search string

    if( substring.charAt(subStringIndex) == search.charAt(searchIndex) ){

        return isSubSequence(substring, search, subStringIndex + 1,
                                                        searchIndex + 1);
    }
    else { //recursive call

        return isSubSequence(substring, search, subStringIndex,
                                                        searchIndex + 1);
    }
}
```

(b) **Tracing** (12 points) Given the following recursive function:

```
public static void mystery(int x, int y) {
    System.out.print(x + " " + y);
    if (x < 0) {
            mystery(-x, y);
    } else if (y < 0) {
            mystery(x, -y);
    } else if (x < 10) {
        System.out.print(" DONE ");
    } else {
        System.out.print("[");
        mystery(x / 10 + y % 10, x % y);
        System.out.print(x + y + "]");
    }
}
```

i. What will be printed when running `mystery(61, 22);` (3 points)

`61 22[8 17 DONE 83]`

ii. What will be printed when running mystery(104, -43); (3 points)

`104 -43104 43[13 18[9 13 DONE 31]147]`

ii.a. How many recursive calls were executed (after the original method call)? (3 points)

`3`

ii.b List the parameters of each recursive call (3 points)

`(104, 43) (13, 18) (9, 13)`

6. **Iteration (12 points)**

After staying up all night answering piazza questions, Michael and the rest of his TA friends were very tired when writing a program to demonstrate in recitation, and had a few bugs.

For example, given `{1, 3, 4, 5, 6, 7, 8}` and 3 as our inputs, we return the array: `{0, 7, 3}` as 0 numbers are divisible by 0, 7 numbers are divisible by 1, and 3 numbers are divisible by 2.

If given `{12, 3, 5, 15, 5}` and 6, we return `{0, 5, 1, 3, 1, 3}`.

The program takes in an integer array, and an integer which represents the length of our return array. For each of the return array's indices, we count how many numbers in the original array are divisible by that index. We assume that 0 numbers are divisible by 0.

Can you help them fix the code before they have to show it during recitation (Fill in the blanks)?

```
public static int[] divisibleArray(int[] arr, int outLength) {

      int[] output = new int[outLength];
      output[0] = 0;
      for( int i = 1; i < output.length; i++) {
          for(int j = 0; j < arr.length; j++) {

              if(arr[j] % i == 0) {

                    output[i]++ ;
              }
          }
      }
      return output;
  }
```

7. **Sorting: Insertion Sort Puzzle(12 points)**

   You just learned about sorting and you excitedly copied the `insertion sort` code from the lecture. But some people who are jealous of you jumbled up all the lines of code.

   Write the full code in the space given below. You may not use any lines/code other than what is given to you. However, you may use any number of opening/closing curly braces ({}).

```
public static  void insSort(double[] A)
        (1) for (int j=i; j>0; j--)
        (2) A[j-1] = tmp;
        (3) for (int i=1; i<A.length; i++)
        (4) A[j] = A[j-1];
        (5) double tmp = A[j];
        (6) if(A[j] < (A[j-1])
}
```

```
public static  void insSort(double[] A){
         for (int i=1; i<A.length; i++){
           for (int j=i; j>0; j--){
             if(A[j] < (A[j-1]){
               double tmp = A[j]; A
               A[j] = A[j-1];
               A[j-1] = tmp;
             }
           }
         }
}
```

8. **Coding (15 points)**

Two of the Head TAs were wondering how many of the guests to the CIS110 party have names that start with specific letters. Specifically, one of the head TAs thinks that more of the guests will have names starting with the letters `'e', 'r', 'i', or 'c'`, but the other one thinks more of the guests will have names starting with `'f', 'o', 'u', or 'h'`. Given a String array containing names of the guests, write a function `ericVsFouh(String[] guestList)` that **returns `true` if there are strictly more names in guestList starting with `'e', 'r', 'i', or 'c'` than names starting with `'f', 'o', 'u', or 'h'`.**

You may assume that all Strings in guestList are lowercase and non-null.

Hint: You may find the function `String.charAt(x)` useful. **Your function does not need to be recursive**

Examples:

If guestList is `{"hanna", "ryan", "eddie", "alice"}`,

`ericVsFouh(String[] guestList)` should return `true` as 2 names start with `'e', 'r', 'i', or 'c'` and 1 names starts with `'f', 'o', 'u', or 'h'`.

If guestList is `{"olivia", "bob", "isabella", "peter"}`,

`ericVsFouh(String[] guestList)` should return `false` as only 1 name starts with `'e', 'r', 'i', or 'c'` and 1 name start with `'f', 'o', 'u', or 'h'`.

```
public static boolean ericVsFouh(String[] guestList) {
        int ericCount = 0;
        int fouhCount = 0;
        for (int i = 0; i < guestList.length; i++) {
            char c = guestList[i].charAt(0);
            if (c == 'e' || c == 'r' || c == 'i' || c == 'c') {
                ericCount++;
            } else if (c == 'f' || c == 'o' || c == 'u' || c == 'f') {
                fouhCount++;
            }
        }
        return ericCount > fouhCount;
    }
```

Feel free to use this page as scratch paper. (If you write anything here that you want us to grade, make sure you clearly indicate this in the answer area earlier in the exam.)