CS 61B Spring 2023

MT1 Exam Review Session

February 3, 2023

Note this worksheet is very long and is not expected to be finished during the session.

1 The Reptile Room

Write out the what the program will output.

```
1  public class Reptile {
2      public String type;
3      public String name;
4      public static String location;
5      public int age;
6
7      public Reptile(String type, String name, String location, int age) {
8          this.type = type;
9          this.name = name;
10         this.location = location;
11         this.age = age;
12     }
13
14     public static void relocate(String l) {
15         location = l;
16     }
17
18     public static void birthday(Reptile a) {
19         a.age += 1;
20     }
21
22     public static void swap(Reptile a, Reptile b) {
23         String temp = a.type;
24         a.type = b.type;
25         b.type = temp;
26     }
27
28     public static void flop(Reptile a, Reptile b) {
29         Reptile temp = a;
30         a = b;
31         b = temp;
32     }
33
34     public static void main(String[] args) {
35         Reptile a = new Reptile("Iguana", "Isabella", "North Carolina", 3);
36         Reptile b = new Reptile("Snake", "Katya", "Colorado", 5);
37         System.out.println(a.location);
```

```
38          Reptile c = new Reptile("Crocodile", "Suha", "California", 1);
39          System.out.println(a.location);
40          System.out.println(Reptile.location);
41          Reptile d = new Reptile("Gator", "Ram", "Georgia", 6);
42          System.out.println(b.location);
43          relocate("Alaska");
44          System.out.println(c.location);
45          System.out.println(d.location);
46          System.out.println(Reptile.location);
47          birthday(a);
48          System.out.println(a.name+" the "+a.type+" turned "+String.valueOf(a.age)+" in "+a.location+"
49          !");
50          flop(c, b);
51          System.out.println(b.type);
52          swap(d, c);
53          System.out.println(d.type);
54      }
55  }
```

2 Reverse Reverse

Implement the `reverse` method below that reverses the ArrayList destructively.

```
\** Return's the value at index. *\
public Object get(int index)

\** Set's the value at index to be value. *\
public void set(int index, Object value)
```

You may use the methods above.

```
1  public ArrayList<Integer> reverse(ArrayList<Integer> alist) {
2
3      for ( ___________________; _________________ / 2; _________________ ) {
4
5          int temp = ____________________________________;
6
7          ____________________________________________;
8
9          ____________________________________________;
10     }
11     return ____________________________________________;
12 }
```

3 ARRAYana grande

After executing the code, what are the values of Foo in `xx` and `yy`?

```
1  public class Foo {
2      public int x, y;
3
4      public static void main(String[] args) {
5          int N = 3;
6          Foo[] xx = new Foo[N], yy = new Foo[N];
7
8          for (int i = 0; i < N; i++) {
9              Foo f = new Foo();
10             f.x = i;
11             f.y = i;
12             xx[i] = f;
13             yy[i] = f;
14         }
15
16         for (int i = 0; i < N; i++) {
17             xx[i].y *= 2;
18             yy[i].x *= 3;
19         }
20
21     }
22 }
```

4 Transposing a 2D Triangular Array

The transpose of a 2D array is a new 2D array whose rows are the columns of the original (and vice versa). In a triangular 2D array, you are guaranteed that every row has exactly one less element than the row above it. The last row is guaranteed to have one element. For example, let `A` be the 2D array below on the left. The transpose of `A` is the resulting 2D array below on the right.

Extra: What if we were only guaranteed that the length of each row is less than or equal to the size of the row above it (not always 1 smaller). What would you change about the algorithm?

$$A = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline 5 & 6 & 7 \\ \cline{1-3} 8 & 9 \\ \cline{1-2} 10 \\ \cline{1-1}\end{array} \implies \begin{array}{|c|c|c|c|}\hline 1 & 5 & 8 & 10 \\ \hline 2 & 6 & 9 \\ \cline{1-3} 3 & 7 \\ \cline{1-2} 4 \\ \cline{1-1}\end{array} = transposeTriangular(A)$$

Given a triangular 2D array `A`, non-destructively transpose `A`.

```
/**
 * Non-destructively transposes A. Assume that A is triangular.
 */
public static int[][] transposeTriangular(int[][] A) {

    if (_____________________________________________________) {

        return ___________________________________________
    }

    ____________________________________________________________

    for (_____________________________________________________) {

        ____________________________________________________________

        ____________________________________________________________

        _____________________________________________________ {

            ____________________________________________________
        }
    }

    return ____________________________________________________
}
```

hey yaal this is vidya and i just wanted to say YAAL GOT THIS!

5 Rocks Rock!

For each line in the `main` method below, write what, if anything, is printed after its execution. Write CE if there is a compiler error and RE if there is a runtime error. If a line errors, continue executing the rest of the lines.

```
1  class Rock {
2      static boolean earthMaterial = true;
3      int type;
4
5      public Rock(int type) {
6          earthMaterial = !earthMaterial;
7          this.type = type;
8      }
9
10     public static void changeMaterial(boolean newBool) {
11         earthMaterial = newBool;
12     }
13
14     public static void main(String[] args) {
15         Rock stone = new Rock(12);
16         Rock spaceRock = new Rock(349);
17         changeMaterial(false);
18         stone.changeMaterial(true);
19         System.out.println(stone.type);
20         System.out.println(spaceRock.type);
21         System.out.println(spaceRock.earthMaterial);
22         System.out.println(Rock.earthMaterial);
23         System.out.println(Rock.type);
24     }
25 }
```

6 Dogs!

Suppose we have the `RowdyDog` interface and the `Dog` and `Corgi` classes below.

```
1  class Dog {
2      public void bark() {
3          System.out.println("Woof");
4      }
5      public void eat() {
6          System.out.println("Yay");
7      }
8  }
9
10 interface RowdyDog {
11     void bark();
12     default void loudBark() {
13         System.out.println("WOOF");
14     }
15 }
16
17
18 class Corgi extends Dog implements RowdyDog {
19     public void bark() {
20         System.out.println("Aroo");
21     }
22     public void angryBark() {
23         System.out.println("Rawr");
24     }
25 }
```

For each line below, write what, if anything, is printed after its execution. Write CE if there is a compiler error and RE if there is a runtime error. If a line errors, continue executing the rest of the lines.

```
Corgi jake = new Corgi();
Dog milton = new Dog();
jake.bark();

jake.eat();

jake.loudBark();

jake.angryBark();

milton.bark();

milton.loudBark();
```

7 More Dogs!

Suppose we have the `Dog` and `HungryDog` classes below.

```
class Dog {
    public void eat() {
        System.out.println("Eating !");
    }

    public void eat(String food) {
        System.out.println("Eating " + food);
    }
}

class HungryDog extends Dog {
    public void eat() {
        System.out.println("Eating a lot !");
    }

    public void bark() {
        System.out.println("Bark !");
    }
}
```

For each line below, write what, if anything, is printed after its execution. Write CE if there is a compiler error and RE if there is a runtime error. If a line errors, continue executing the rest of the lines.

```
Dog a = new Dog();
Dog b = new HungryDog();
HungryDog c = new Dog();
HungryDog d = new HungryDog();

a.eat();

a.eat("dog food");

d.eat();

b.eat();

b.eat("dog food");

a.bark();

b.bark();

d.bark();
```

8 Dedup (SP17 MT1) *Extra*

Fill in the blanks to implement the `removeDuplicates` method correctly

Given a sorted linked list of items, remove the duplicates. For example, given `1 -> 2 -> 2 -> 2 -> 3`, mutate it to become `1 -> 2 -> 3` destructively.

```
public class IntList {
    public int first;
    public IntList rest;

    public static void removeDuplicates(IntList p) {

        if (_____________________________________________) {
            return;
        }
        IntList current = _____________________________________________;

        IntList previous = _____________________________________________;

        while (_________________________________________) {

            if (current.first == _________________________________________) {

                previous.rest = _________________________________________;
            } else {

                _____________________________________________;
            }

            _____________________________________________;
        }
    }
} // Reminder: Only write one statement per line.
```