CS 61B Guerrilla Section 1 Worksheet

4 February 2017

Directions: In groups of 4-5, work on the following exercises. Do not proceed to the next exercise until everyone in your group has the answer and *understands why the answer is what it is*. Of course, a topic appearing on this worksheet does not imply that the topic will appear on the midterm, nor does a topic not appearing on this worksheet imply that the topic will not appear on the midterm.

1 Grandpuppies!

Given the following block of code, answer the following questions.

```
1  public class Dog {
2      public String name;
3
4      public Dog(String name) {
5          this.name = name;
6      }
7
8      public Dog giveBirth() {
9          return new Dog(this.name + "'s puppy");
10     }
11
12     public void bark() {
13         System.out.println(this.name + " barks!");
14     }
15
16     public static void main(String args[]) {
17         Dog[] myDogs = new Dog[3];
18         /* YOUR CODE HERE */
19     }
20 }
```

(a) Given the above code, what would you write in the main method to populate `myDogs` with 2 new Dogs named Fido and Fiddle?

(b) How would you make Fido's grand-child (the puppy of Fido's puppy) bark, in only one line of code?

(c) What would your answer to (b) output?

(d) What would happen if we tried to call `myDogs[2].bark()`?

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

2 Knapsack

Fix the bugs in the Knapsack class below, so that main prints out "Doge coin:100.45"

```
1  class Knapsack {
2      public String thing;
3      public String amount;
4
5      public Knapsack(String str, double amount) {
6          String thing = str;
7          amount = amount;
8      }
9
10     public Knapsack(String str) {
11         Knapsack(str, 100.45);
12     }
13
14     public static void main(String[] args) {
15         Knapsack sack = new Knapsack("Doge coin");
16         System.out.println(thing + " : " + amount);
17     }
18 }
```

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

3 Cat World Domination

Toby wants to rule the world with cats. To do this, he's built a `Cat` class to start building his army. In his army of cats, there is a family hierarchy where each cat may or may not have only one parent, and may or may not have kitties (stored in the form of an `ArrayList`). Each cat that has a parent is also a kitty of that parent.

To speed up the world domination process, Toby builds a method called `copyCat` that, in addition to copying the cat, also copies of that cat's descendants. Toby tries to copy his cats initially, but realizes it doesn't work the way he expects it to. Here is his `Cat` class:

```
1  public class Cat {
2      private Cat parent;
3      private ArrayList<Cat> kitties;
4      private String name;
5
6      public Cat(Cat parent, String name) {
7          this.name = name;
8          this.kitties = new ArrayList<Cat>();
9          this.parent = parent;
10     }
11
12     public Cat copyCat() {
13         Cat copy = new Cat(this.parent, this.name);
14         for (int i = 0; i < this.kitties.size(); i += 1) {
15             copy.kitties.add(this.kitties.get(i).copyCat());
16         }
17         return copy;
18     }
19 }
```

What's wrong with his Cat class? Drawing a box and pointer diagram may help!

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

4 Samehorse

```
1  public class Horse {
2      Horse same;
3      String jimmy;
4
5      public Horse(String lee) {
6          jimmy = lee;
7      }
8
9      public Horse same(Horse horse) {
10         if (same != null) {
11             Horse same = horse;
12             same.same = horse;
13             same = horse.same;
14         }
15         return same.same;
16     }
17
18     public static void main(String[] args) {
19         Horse horse = new Horse("you've been");
20         Horse cult = new Horse("horsed");
21         cult.same = cult;
22         cult = cult.same(horse);
23         System.out.println(cult.jimmy);
24         System.out.println(horse.jimmy);
25     }
26 }
```

What would Java display? Draw a box-and-pointer diagram.

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

5 Arrays

```
class Foo {
    int x;
    int y;
}
public class ArraysQuestion {
    public static void main(String[] args) {
        int N = 3;
        Foo[] xx = new Foo[N];
        Foo[] yy = new Foo[N];
        for (int i = 0; i < N; i++) {
            Foo f = new Foo();
            f.x = i; f.y = i;
            xx[i] = f;
            yy[i] = f;
        }
        for (int i = 0; i < N; i++) {
            xx[i].y *= 2;
            yy[i].x *= 3;
        }
    }
}
```

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

`xx[0]:` `yy[0]:`

`xx[1]:` `yy[1]:`

`xx[2]:` `yy[2]:`

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

6 Braid

Write `braid`, a method that takes in two `IntDList`s of equal length and interleaves the linked lists and returns the front of the new list. For example, given `a = [1, 2, 3]` and `b = [4, 5, 6]`, it should return `[1, 5, 3, 4, 2, 6]`.

```
public class IntDList {
    public int item;
    public IntDList prev, next;

    public static IntDList braid(IntDList A, IntDList B) {









    }
}
```

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

7 Triangularize

Write `triangularize`, a method that takes in an array of `Intlists R` and a single `Intlist L`, and breaks `L` into smaller `Intlists`, storing them into `R`. The `Intlist` at index k of `R` has at most $k+1$ elements of `L`, in order. thus concatenating all of the `Intlists` in `R` together in order would give us `L` back. Assume `R` is big enough to do this. For example, if the original `L` contains `[1, 2, 3, 4, 5, 6, 7]`, and `R` has 6 elements, then on return `R` contains `[[1], [2, 3], [4, 5, 6], [7], [], []]`. If `R` had only 2 elements, then on return it would contain `[[1], [2, 3]]`. `triangularize` may destroy the original contents of the `Intlist` objects in `L`, but does not create any new `Intlist` objects. Note: assume `R`'s items are all initially `null`.

```
1  public static void triangularize(IntList[] R, IntList L) {
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25 }
```

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

8 Sorting Zeros and Ones (Extra for Experts)

Write `sortZerosOnes`, a method that takes an `Intlist` of only 0's and 1's and sorts the nodes of the list, and returns the new start node. Do not change any head values or create any new nodes. You can assume that the list `L` passed in has only 0's or 1's.

```
1  public static IntList sortZerosOnes(IntList L) {
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25 }
```

STOP!

Don't proceed until everyone in your group has finished and understands all exercises in this section!

9 2-D Arrays (Extra for Experts)

(a) Write `diagonalFlip`, a method that takes a 2-D array `arr` of size $N \times N$ and **destructively** flips `arr` along the diagonal line from the left bottom to right top.

```
public static void diagonalFlip(int[][] arr) {




}
```

(b) Write `rotate`, a method that takes a 2-D array `arr` of size $N \times N$ and **destructively** rotates `arr` 90 degrees clockwise.

```
public static void rotate(int[][] arr) {




}
```