CS 61B Summer 2020 | Small Group Tutoring Midterm 1 Review Section

MT 1 Review Worksheet

1 Pass By Value

Spring 2015 MT 1

```
public class Foo {
    public int x, y;
    public Foo (int x, int y) {
        this.x = x;
        this.y = y;
    }
    public static void switcheroo (Foo a, Foo b) {
        Foo temp = a;
        a = b;
        b = temp;
    }
    public static void fliperoo (Foo a, Foo b) {
        Foo temp = new Foo(a.x, a.y);
        a.x = b.x;
        a.y = b.y;
        b.x = temp.x;
        b.y = temp.y;
    }
    public static void swaperoo (Foo a, Foo b) {
        Foo temp = a;
        a.x = b.x;
        a.y = b.y;
        b.x = temp.x;
        b.y = temp.y;
    }
    public static void main(String[] args) {
        Foo foobar = new Foo(10, 20);
        Foo baz = new Foo(30, 40);
        switcheroo(foobar, baz);
        // What are the values of foobar.x, foobar.y, baz.x, baz.y
        // 10, 20, 30, 40
        fliperoo(foobar, baz);
        // What are the values of foobar.x, foobar.y, baz.x, baz.y
        // 30, 40, 10, 20
        swaperoo(foobar, baz);
        // What are the values of foobar.x, foobar.y, baz.x, baz.y
        // 10, 20, 10, 20
    }
```

```
}
```

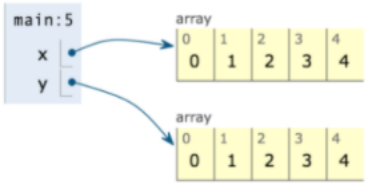
Write the contents of the foobar and baz variables at the indicated points in time.

2 Pass By Value and Arrays

Spring 2015 MT 1

Our hero Bilbo tries to execute the code below, and finds to his surprise that the two arrays are not considered equal. Consulting Head First Java, he reads that == returns true "if two references refer to the same object". He remembers from lecture that an array consists of a length and N simple containers (where N = length), and reasons (correctly) that this means that the underlying objects pointed to by x and y must be different. He then runs it through the visualizer and observes the figure to the right.

```
public static void main(String[] args) {
int[] x = new int[]{0, 1, 2, 3, 4};
int[] y = new int[]{0, 1, 2, 3, 4};
System.out.println(x == y);
}
```



Given what Bilbo has learned while debugging his code, for each of the following, answer whether the code will print true, print false, or state that there is not enough information. Please use the three blanks provided to the right of each print statement. Assume that the code compiles and executes without error.

```
public static void main(String[] args) {
int[] x = new int[]{0, 1, 2, 3, 4};
int[] y = new int[]{0, 1, 2, 3, 4};
y = someUnknownFunction(x, y);
System.out.println(x == y);                    not enough information
}
public static void main(String[] args) {
int[] x = new int[]{0, 1, 2, 3, 4};
int[] y = new int[]{0, 1, 2, 3, 4};
anotherUnknownFunction(x, y);
System.out.println(x == y);                   false
}
public static void main(String[] args) {
int[] x = new int[]{0, 1, 2, 3, 4};
int[] y = new int[]{0, 1, 2, 3, 4};
System.arraycopy(x, 0, y, 0, 5);
System.out.println(x == y);                    false
}
```

3 Arrays

Spring 2016 MT 1 You are a programmer for the people's glorious state of Oceania. Your citizens have started trying to avoid censorship by rotating their text messages. For example, suppose a citizen wants to use the banned word "light", for example: "nice light today". Instead, the citizen might say "ght today nice li". It is your job to write code to remove banned words. Throughout this problem you may assume that message and banned are of length greater than zero, and that banned.length <message.length. You may also assume that k is a valid number between 0 and message.length - 1. Hint: for positive integers a % b returns the remainder of dividing a by b, e.g. 7 % 3 is 1.

Fill in the matches method below, which returns true if the message starting at k matches the banned word, treating the message as circular. For example, if message is ['d', 'd', 'o', 'g', 'b', 'a'] and banned is ['b', 'a', 'd'], this method will return true for k = 4, and false for all other k. You may not need all lines. If you don't use a line, leave it blank.

```
private static boolean matches(char[] message, char[] banned, int k) {
        ____________________________________________________
        for (int i = __0__; i < ___banned.length___; i += 1) {
        int messageIndex = ___(k + i) % message.length____;
        if (message[messageIndex] != banned[i]) {
            return false_____________________;
        }
        }
        _____return true_____________________________;
}
```

Complete the helper function matchStart, which returns the start index of a banned word in a message. For example, if message is ['d', 'd', 'o', 'g', 'b', 'a'] and banned is ['b', 'a', 'd'], this method should return 4, because the match begins at position 4. If there is no match, return -1. You can do this part even if you skipped part a! You may use the method from part a, even if you didn't actually get it right. You may assume that banned words occur only once (if at all).

```
public static int matchStart(char[] message, char[] banned) {
        for (int k = 0; k < message.length; k += 1) {
            if (matches(message, banned, k)) {
                return k;
            }
        }
        return -1;
}
```

4 IntLists

Spring 2017 MT 1 Fill in the blanks below to correctly implement removeDuplicates.

```
public class IntList {
    public int item;
    public IntList next;
    public IntList (int f, IntList r) {
        this.item = f;
        this.next = r;
    }
    /**
     * Given a sorted linked list of items - remove duplicates.
     * For example given 1 -> 2 -> 2 -> 2 -> 3,
     * Mutate it to become 1 -> 2 -> 3 (destructively)
     */
    public static void removeDuplicates(IntList p) {
        if (p == null) {
            return;
        }
        IntList current = p.next;
        IntList previous = p;
        while (current != null) {
            if (current.item == previous.item) {
                previous.next = current.next;
            } else {
                previous = current;
            }
            current = current.next;
        }
    }
}
```

5 DLLists

Summer 2018 MT 1

Implement DLList.duplicate, which destructively copies each node in the list, resulting in a list that's twice as long. For example, if the list initially contained [1, 2, 3], it should contain [1, 1, 2, 2, 3, 3] after the call to duplicate.

```
public class DLList<BleepBlorp> {
    private class Node {
        public Node prev;
        public BleepBlorp item;
        public node next;
        public Node(BleepBlorp i, Node p, Node n) {
            item = i;
            prev = p;
            next = n;
        }
    }
    private Node sentinel;
    private int size;
    public void duplicate() {
        Node p = sentinel.next;
        while (p != sentinel ) {
            Node copy = new Node(p.item, p, p.next );
            copy.prev.next = copy ;
            copy.next.prev = copy ;
            p = copy.next ;
        }
        size *= 2;
    }
}
```

6 Deques

On project 1, you implemented the Deque interface as ArrayDeque and LinkedListDeque. In this problem, you'll improve your Deque interface by adding a new default method. Your job: Add a default method remove(Item x) that removes all items equal to x. This method should return true if anything was removed. It should remove false if nothing was removed. You may not use the new keyword. Only write one statement per line. You may not need all the lines. If you need to determine that two Items are equal, use the .equals method, i.e. don't use == to compare Items.

```
public interface Deque<Item> {
    void addFirst(Item x); void addLast(Item x);
    boolean isEmpty(); int size();
    void printDeque(); Item get(int index);
    Item removeFirst(); Item removeLast();
    default boolean remove(Item x) {
        int length=size();
        for(int i=0;i<length; i++){
            Item first=removeFirst();
            if(!((first!=null&&first.equals(x))||first==x)){
                addLast(first);
            }
        }
        return length!=size();
    } /* Answers using the new keyword will not be given credit. */
}
For example, the test below should pass:
Deque<Dog> dd = new ArrayDeque<Dog>();
dd.addLast(new Dog(\Ljilja"));
dd.addLast(new Dog(\Rikhard"));
dd.addLast(new Dog(\Spartacus");
dd.addLast(new Dog(\Rikhard"));
assertEquals(4, dd.size());
boolean rikhardRemoved = dd.remove(\Rikhard");
assertTrue(rikhardRemoved);
assertEquals(2, dd.size());
```

Reminder: You may not use the new keyword! This is not just an arbitrary restriction, but ensures that we don't use any unnecessary space.