Midterm 1 Review Document

CS 61B Spring 2017

Antares Chen + Kevin Lin

Introduction

This document is meant to provide you supplementary practice questions for the upcoming midterm. It reflects on material that may have already seen in labs and lecture. Do not use this as a “be all end all” guide! It is still highly recommended that you review previous and external course material.

First I would recommend you read through and maybe take notes on the labs and quizzes we’ve done up until now. Remember that the tests we write will take from material that you’ve seen there and on your quizzes! Your textbook will also be useful for learning some of this material.

After reviewing the material, try to find previous semester midterms and practice the content on given on them. Websites such as that hosted by HKN will be very useful.

Finally, make sure you don’t stress! This class is hard and will only be made harder if you stress yourself out. But you smart, you loyal #blessup. Ask lots of questions and let us the TA’s help you (believe me when I say we want you guys to succeed).

If you find yourself beginning to panic, simply pause, breathe, and believe. In the off chance you don’t believe in yourself, then believe in me who believes in you.

Know Your Java

This Code Sucks!

This code sucks… no seriously some of the below code examples are broken. For this question, we will be looking at the following classes.

```java
public class A {
    private int valA = 2;
    public void f() {
        this.g();
    }
    public void g() {
        System.out.println("A:" + valA);
    }
    public int h() {
        return valA;
    }
    public static A createA() {
        return new A();
    }
}
```

```java
public class C extends B {
    private int valC = 42;
    public void f() {
        this.g();
    }
}
```

```java
public class B extends A {
    private int valB = 15;
    protected int value = 1337;
    public void g() {
        System.out.println("h:" + h() + " z:" + valB);
    }
    public void banana() {
        System.out.println("no");
    }
}
```

Beneath are a number of coding examples. Next to each, determine if the code can run. If it can, write down the output, else state what type of error (compile-time vs runtime) is thrown and for what reason.

`// to be placed in class A`
`public static void main(String[] args) {`
`    A this = new A();`
`    this.g();`
`}`	

`// to be placed in class A`
`public static void main(String[] args) {`
`    A thing = this.createA();`
`    thing.f();`
`}`	

`// to be placed in class A`
`// what is this?`
`public A(int valA) {`
`    this.valA = valA;`
`}`	

`// to be placed in class A`
`public static void main(String[] args) {`
`    A thingC = new C();`
`    System.out.println(thingC.valC);`
`}`	

`// to be placed in class A`
`public static void main(String[] args) {`
`    A thingB = new B();`
`    thingB.banana();`
`}`	

`// to be placed in class A`
`public static void main(String[] args) {`
`    A thingB = new B();`
`    thingB.g();`
`}`	

`// to be placed in class B`
`public static void main(String[] args) {`
`    A thingB = new B();`
`    System.out.println(thingB.valA);`
`}`	

`// to be placed in class B`
`public static void main(String[] args) {`
`    B thingB = new B();`
`    System.out.println(thingB.h());`
`}`	

`// to be placed in class B`
`public static void main(String[] args) {`
`    System.out.println(this.valB);`
`}`	

`// to be placed in class B`
`public static void main(String[] args) {`
`    System.out.println(B.valB);`
`}`	

`// to be placed in class C`
`public static void main(String[] args) {`
`    C thingC = new C();`
`    System.out.println(thingC.value);`
`}`	

`// to be placed in class C`
`public static void main(String[] args) {`
`    C thingC = new A();`
`    System.out.println(thingC.valC);`
`}`	

What Type is This?

Look at the code blocks below. In the box besides it, write down the highlighted variables static and dynamic type and then write down what the statement would execute to.

```java
public class Main {
    public static void method(B arg1) {
        System.out.println(arg1.g());
    }

    public static void main(String[] args) {
        C thingC = new C();
        method(thingC);
    }
}
```

```java
public class Main {
    public static void method(A arg1) {
        C arg2 = (C) arg1;
        arg2.f();
    }

    public static void main(String[] args) {
        C thingC = new C();
        method(thingC);
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        A thingC = new C();
        thingC.f();
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        A thing = new B();
        B thingy = (B) thing;
        thingy.banana();
    }
}
```

Functions as Objects

Unlike Python, Java doesn't treat functions as first class objects. This means that we can't simply pass methods into each other in a functional manner. However, we can employ object orientation to get around this. Consider the following class

```
/** IntFunction is an object representing functions that return ints. */
public class IntFunction {
    // an instance variable that represents an incoming argument.
    int arg;
    public void setArg(int arg) {
        this.arg = arg;
    }
    public int apply() {
        return -1;
    }
}
```

We can consider `IntFunction` as a base class for all functions that accepts an int and returns an int. The actual functionality of the method we wish to represent would go into the instance method apply. By default, apply returns -1. To produce more complicated behavior (such as methods that accept arguments) we can simply extend `IntFunction`. Consider `SquareFunction`, a method that squares an input argument.

```
/** SquareFunction represents a functions that squares ints. */
public class SquareFunction extends IntFunction {
    public int apply() {
        return arg * arg;
    }
}
```

And finally, we can now apply it in a functional manner!

```
public static void main(String[] args) {
    int[] inputs = {1, 2, 3, 4, 5};
    SquareFunction sf = new SquareFunction();
    for (int i = 0; i < inputs.length; i += 1) {
        sf.setArg(inputs[i]);
        inputs[i] = sf.apply();
    }
    // now inputs = {1, 4, 9, 16, 25};
}
```

1) Define an object representing the `isMultipleOf` function. That is your object should be able to take in an integers and N, returning the number if it is a multiple of N else returning -1.

```
public class ____________________________________ {
    // what do I need as instance variables?


    // wait there's no return type...
    public ________________(__________) {
        _____________;
    }

    public void ___________(__________) {
        _____________;
    }

    // what else do I need?













}
```

2) Define the map function in this main class. The map function should take in any functional object and apply it to each element of an int list. **NOTE** for easy-mode write map iteratively, for *hard-mode* write map recursively

```
public class Main {

    public static void easyMap(______________, _______________) {
        for (_________________________________________) {
            _________________________________;
            _________________________________;
        }
    }

    public static void hardMap(______________, _______________) {
        _____________________________________;
    }

    // helper functions may go here.













}
```

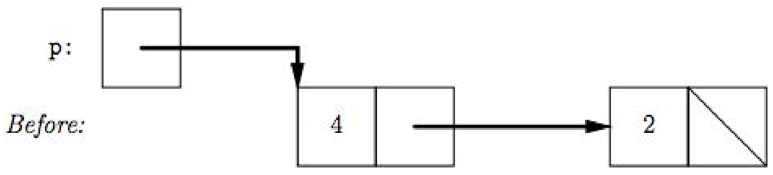
DBCCCCCCCCCCC

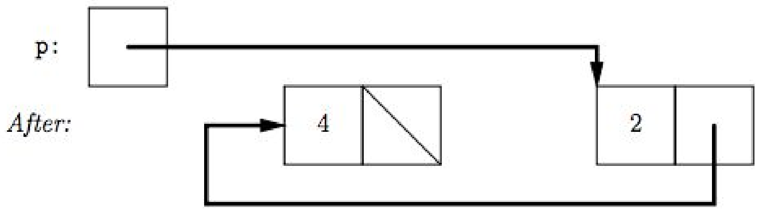
Don’t be clueless and read the documentation below! Then write the corresponding code that implements the functionality.

```
public class DBC {

    public static void main(String[] args) {
        /* 2. Declare a variable named "a" of type int. */
        _________________________________________;

        /* 3. Assign the value 3 to a.*/
        _________________________________________;

        /* 4. Declare another variable named b of type
         * int, initializing with value 5. */
        _________________________________________;

        /* 5. Declare a variable named c and initialize
         * it with the result of invoking the
         * sum method on a and b. */
        _________________________________________;

        /* 6. Declare a variable named chaka, which is
         * an array of bytes, with size 7. */
        _________________________________________;

        /* 7. Declare a variable named khan, which is
         * an array of longs, with contents 1, 2, 3, 4,
         * 5. */
        _________________________________________;

        /* 8. Set khan's second item (index 1) as chaka's
         * last item. */
        _________________________________________;

        /* 9. You are given the following array with unknown
         * size. Write a for-loop that prints all elements
         * IN REVERSE */
        int[] someArray = ...;
        _________________________________________________________
        _________________________________________________________
        _________________________________________________________
        /* 10. You are given an integer n. Using a for
         * loop, create an n by n identity matrix called
         * eye, where all diagonal entries are set to 1
         * and other entries are set to 0 from top left
         * to bottom right. You can represent the matrix
         * as an array of int arrays. Hint: the default
         * value for int is 0. */
        int n = ...;
        ________________________________________________________
        ________________________________________________________
        ________________________________________________________
        ________________________________________________________
        ________________________________________________________
    }
}
```

Pox and Bointers

The before picture shows the state of an `IntList` object and one local variable. In the lines provided, write the Java code statements that transforms the before picture to the after, subject to the following restrictions. (1) do not modify the `first` instance variable and (2) do not introduce any new variables.





____;
____;
____;

Arrays

Easy Mode

These questions test basic understanding of manipulating arrays and such.

Hella Deep Copy Given a 3D array, make a deep copy of the array. That is return a new 3D array that contains the same elements as the previous. Remember the array may be ragged!

```
public static int[][][] hellaDeepCopy(int[][][] array) {









}
```

Print All Given a 2D array, print all the elements. Remember it's possible that the array is ragged! Make sure to line break each int[] iterated through.

```
public static void printAll(int[][] array) {









}
```

Medium Mode

This section provides questions that should match your understanding of arrays after labs.

Iterative Fibonacci Array Given a number *N* return an array of arrays such that the ith array contains all fibonacci numbers up to the ith one. For example if the *N*=4, the return result would be `[[1], [1, 1], [1, 1, 2], [1, 1, 2, 3]]`.

```
public static int[][] fibonacciArray(int n) {
    int[][] result = _______________________;
    for (int i = _____; _____________; ___________) {
        result[i] = __________________;
        for (int j = _____; ____________; ___________) {
            if (______________ || _____________) {
                result[i][j] = 1;
            } else {
                result[_][_] = ____________________;
            }
        }
    }
    return result;
}
```

IntList to Array convert an `IntList` to an array.

```
// assume that IntList has a size() method returning the length of the list
public static int[] toArray(IntList list) {
    int[] ____________ = ________________;
    toArrayHelper(___________, ___________, ___________);
    return ____________;
}

public static void toArrayHelper(int index, int[] nList, IntList list) {
    if (list == _______) {
        _______________________________;
    } else {
        nList[_________] = ______________;
        _________________________________;
    }
}
```

Hard Mode

This last section contains non-trivial, purposefully obfuscated applications of arrays. Try your best!

Recursive Fibonacci Array Do the fibonacci array question again, but recursively!

```
public static int[][] fibonacciArray(int n) {
    int[][] results = ___________________;
    ___________________________;
    return results;
}

public static void fibArrayHelper1(int[][] something, int k) {
    if (___________________) {
        something_____ = _____________________________;
        ___________________________________;
        ___________________________________;
    }
}

public static void fibArrayHelper2(int[] array, int k) {
    if (__________________) {
        array[k] = 1;
    } else if (k == 1) {
        fibArrayHelper2(array, k - 1);
        array[k] = 1;
    } else {
        ________________________________________;
        ________________________________________;
        array[k] = _________________________;
    }
}
```

Linked Lists

Easy Mode

The next couple of questions will have some linked list questions. The first few will focus on basic understanding of linked lists.

Deep Copy given an `IntList`, return an exact copy of that linked list.

```
public static IntList deepCopy(IntList list) {







}

// in case you want helper methods
```

Find Max Given a `IntList` return the max element of that list.

```
public static int maxElement(IntList list) {







}
```

Double Deep Copy given a non-circular DLList with no sentinel nodes, return an exact copy of that DLList.

```
public static DLList deepCopy(DLList list) {









}

// in case you want helper methods
```

Medium Mode

A few questions that should match on par understanding after lab and practice

Copy Odd Given an `IntList`, non-destructively remove all even elements.

```
public static IntList copyOdd(IntList list) {









}

// in case you want helper methods
```

Palindrome Given a circular DLList with a sentinel node check if it is a palindrome. Your method may be non-destructive. You can assume list and its head are not null.

```
public static boolean palindrome(DLList list) {
    return palindrome(_______________________________________);
}

public static boolean palindrome(DLNode sentinel) {
    if (________________________ == ________________________) {
        return true;
    } else if (sentinel.____________ == sentinel.____________) {
        _____________________________________________________;
        _____________________________________________________;
        return palindrome(sentinel);
    } else {
        _____________________________________________________;
    }
}
```

Asymptotics

Easy Mode

These questions should reflect basic understanding of asymptotics. Reviewing the lab should allow you to do these directly.

Bound me Provide a theta bound for the following equations.

$$f(n) = n^5 + 3n^2 + 12$$

$$f(n) = n\log(n) + n^2$$

$$f(n) = \log_{13}(n)$$

$$f(n) = \log_2(n^5)$$

Discussion time In the following lines we've listed functions f(n) and g(n) as well as if f(n) is in O(g(n)), Omega(g(n)) or Theta(g(n)). State in the blank whether the statement is true and if not provide the correct bound.

$f(n) = 20501$	$g(n) = 1$	$f(n) \in O(g(n))$	____________
$f(n) = n^2 + n$	$g(n) = 0.000001n^3$	$f(n) \in \Omega(g(n))$	____________
$f(n) = 2^{2n} + 1000$	$g(n) = 4^n + n^{100}$	$f(n) \in O(g(n))$	____________
$f(n) = \log(n^{100})$	$g(n) = n\log n$	$f(n) \in \Theta(g(n))$	____________
$f(n) = n\log n + 3^n + n$	$g(n) = n^2 + n + \log n$	$f(n) \in \Omega(g(n))$	____________
$f(n) = n\log n + n^2$	$g(n) = \log n + n^2$	$f(n) \in \Theta(g(n))$	____________
$f(n) = n\log n$	$g(n) = (\log n)^2$	$f(n) \in O(g(n))$	____________

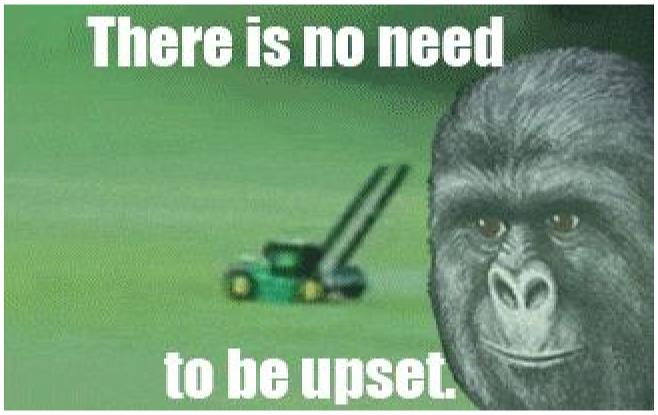
Medium Mode

Here are some asymptotics questions that extend the material you saw in lab. These questions were stolen from fall 2015 midterms and finals!

Final time Give the tightest bound for this expression. (Big O, Omega, Theta)

$$\sum_{i=0}^{k} (\log N)^i N^{k-i}$$

Final time part two! True or false: If $f(N) \in O(N^2)$ and $g(N) \in O(N)$ are positive-valued functions, then $f(N) / g(N) \in O(N)$. If true, explain why; if false give a counterexample.


There is no need
to be upset.