APSI CSA Lecture Notes

June 30, 2022

1 Day 1 (Units 1-4)

- **AP CSA Exam description**
- **AP CSP Exam description**
  - https://www.khanacademy.org/computing/ap-computer-science-principles
  - https://www.helloworldcs.org/curriculum/csp
- **https://codingbat.com/java** (supports Java and Python, very much like AP exam)
  - Ex: "do all of column 1 in String-1", send screenshot but sometimes check for fakes
  - The warmup section is just a warmup to learn the code, don't spend more than 5 minutes on it – otherwise show solution and figure it out, ask questions
- Function Quizzes: Sept-Oct
- Magpie Lab from UW
- CSA - 10 Chapters
- Finish class by March 1st - Onward is review before AP exam (fast model)
- https://code.org/educate/csa
- https://popfizz.io/self-study/ap-computer-science-a
- http://skylit.com/beprepared/fr.html
- https://www.csawesome.org/
- https://codehs.com/
- https://codio.com/
- https://www.zybooks.com/catalog/ap-java/
- https://www.project-stem.com/
- https://www.codingrooms.com/
- https://www.microsoft.com/en-us/makecode
- https://projecteuler.net/
  - Great for advanced students
  - Great for students that finish programs early and "have nothing to do"
- https://zerorobotics.mit.edu/
  - If your school is in the top 15% in the world, you can get the chance to run your code on a robot on the ISS; only ~200 schools compete yearly
- The 2023 AP Exams will be administered May 1–5 and May 8–12
- ~110 assignments in first semester, ~80 assignments second semester
- AP CompSci A CED Walkthrough Tutorial
- 90 mins multiple choice exam and written EACH, 50/50
- Units 5-7 need more weight than the exam overview says
- Don't do much recursion writing in CS A, just reading
- "What did you learn at school today?" - Prep question basically everyday for the first few months

- Start first day with Hello World, keep introductions short
- Used BlueJ because it used to be the only cross-platform IDE
  - If using BlueJ - Options: **Clear screen at method call**, **Unlimited Buffering**
  - Encourage kids to organize BlueJ code into quarter/semester folders
- Put comments on the top that have name, program name, and date to catch cheaters
- Put output of program at bottom of programs
- Know which lines of code are actually being graded per program
- Don't write perfect code all the time; intentionally write errors without running and let the kids catch it
- Make small programs like turning Hello World into Favorite Club/Sport, know the kids
- Primitive data types on exam: `int, double, boolean`
- Don't get kids used to declaring variables without initial values – lose credit on exam
- User Input/Scanner not tested on exam
- Every program you write has the same 3 sections – **Input**, **Calculation**, **Output**
- Math teachers – `myInt = kbReader.nextDouble();` is not legal; also know = vs. ==
- Imports are **never** on the exam
- Don't get into arrays for about 6-8 weeks
- Change variable name for `Scanner input = ...` or kids will think it always have to be called "input"
- Input is always about 3 lines – Prompt, Variable Declaration, Println
- Unit 2 - spend most time on String class
- Properly counting length of string is on exam – substring
- **Suggestive vocab:** "Part of" rather than "Substring", "Otherwise" rather than "Else", "Comes from" for `for (var x : array)` for-each loop, "Is" (or "Points to" for AP 2) rather than "Equals"
- When substring has two arguments, just subtract them to get the new string length
- ~90% of written exams will test double vs. int division, e.g., find the average of (using casting)
- Grading for CS can greatly vary – Getka gives very few exams to combat unweighted grade systems (80% summative, 20% formative)
  - All in-class programs are formative, all homework is summative
  - No deadlines during semester; gradebook dates are meaningless – watch for assignments marked as missing (most kids have this assignment done or it should be done by now)
  - No deadlines helps the teacher teach to high-level kids without burying the strugglers
- If school policy, for advanced students, let them skip intro programming classes and get into AP – by junior/senior year, take 6-9 credits of college classes
- Students get a Java cheatsheet for the exam now
- AP Exam isn't a CompSci exam, it's a programming reading exam
- PEMDAS can trick students when reading, also `ceil()` and `floor()`
  - Keep in mind integer division with ceil/floor, `ceil(3/4)` is **0**
- Use your fingers to count the gap for *random* – (max - min) + 1
- Don't put calc/pre-calc students in introductory classes, put them straight into AP
- Kids will very likely struggle with logical and comparison operators
  - Logical order of operations: Not, And, Or (if no parentheses)
- With multiple if/else-if statements with an else, think about what the else implies about all the conditions before it
  - Hardly ever use **nested** if statements
- Code e iciency **does not matter** on the exam – does the code work?
- Switch/case not on the exam – just use if/else-if

- Never input or output on exam
- Getka doesn't teach to non-AP style because AP graders will grade it quickly (and miss minor errors) – stay on pace with the class and don't divert
  - Jobs will teach you the style they want anyway
- De Morgan's Law is very important for knowing loops
- Never use the ++ operator unless it's on its own line
- Getka uses LCV instead of i for `for` loops because in hand-written code, i looks like a 1
- Lang docs – 100 digit represents the unit (0-99, 100-199, etc.), letter represents difficulty
- For BlueJ, data files have to be in the same directory as the root project directory

1.1 Unit 1 - Primitive Types

[ ]:
```java
// FirstName LastName
// Hello World
// 6/27/2022
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}

// Hello World
HelloWorld.main(null);
```

```
Hello, World!
```

[ ]:
```java
// Full Name
// My Name
// 6/27/2022

public class MyName {
    public static void main(String[] args) {
        System.out.println("Full Name");
    }
}

// Full Name
```

[ ]:
```java
import java.util.*;

// Full Name
// Lang52a
// 6/27/2022

public class Prog52a {
    public static void main(String[] args) {
        // Input Section
        Scanner input = new Scanner(System.in);
```

```
        System.out.print("What is the length of the rectangle: ");
        int len = input.nextInt();
        System.out.println();

        System.out.print("What is the width of the rectangle: ");
        int wid = input.nextInt();
        System.out.println();

        // Calculation Section
        int area = len * wid;
        int perm = len + len + 2 * wid; // Numerous ways to calculate

        // Output Section
        System.out.println("The area of the rectangle is " + area);
        System.out.println("The perimeter of the rectangle is " + perm);
    }
}

// What is the length of the rectangle: 5
//
// What is the width of the rectangle: 6
//
// The area of the rectangle is 30
// The perimeter of the rectangle is 22
```

[ ]:
```
import java.util.*;

// Full Name
// Lang76a
// 6/27/2022

public class Prog76a {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.println("Enter a number you dislike (1-9): ");
        int a = input.nextInt();
        int b = a * 9;
        int c =  b * 12345679;

        System.out.println(a);
        System.out.println("x " + 9);
        System.out.println("____________");
        System.out.println(b);
        System.out.println("x " + 12345679);
        System.out.println("____________");
```

```
        System.out.println(c);
    }
}

// Enter a number you dislike (1-9): 3
// 3
// x 9
// ------------
// 27
// x 12345679
// ------------
// 333333333
```

Note on nextLine():

```
->3
4.5
Cool
```

-> Cursor position when running user input

`myInt = kbReader.nextInt();`

Cursor is now at

```
3->
4.5
Cool
```

`myDouble = kbReader.nextDouble();`

Cursor is now at

```
3
4.5->
Cool
```

`myString = kbReader.nextLine();`

Cursor is at the same spot (read up to enter hit/carriage return), myString is empty.
Need to read in Cool first and it will read correctly

[ ]:
```
import java.util.*;

public class wowsa {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        int a = input.nextInt();
        String junk = input.nextLine(); // Read in line-feed
        // Can use 'input.nextLine();' by itself, but may be confusing
        String wow = input.nextLine();
        String wow2 = input.nextLine();

        System.out.println(wow + wow2);
```

```
    }
}

// 3
// cool
// beans
// coolbeans
```

1.2 Unit 2 - Using Objects

[ ]:
```
public class coolbeans {
    public static void main(String[] args) {
        int a = 10;  // 10
        int b = 3;   // 2
        // int c = a / b; // 5
        // double c = a / b; // 3.0
        // double c = (double)(a / b); // 3.0
        double c = (double)a / b; // 3.3333333333333335
        System.out.println(c);

        String n = "Bob Getka";
        String f = n.substring(0, n.length() / 2);
        String b1 = n.substring(n.length() / 2);
        System.out.println(b1 + f);
    }
}

// 3.3333333333333335
// GetkaBob
```

[ ]:
```
// https://codingbat.com/prob/p115863

public String middleThree(String str) {
    int mid = str.length() / 2;
    int begin = mid - 1;
    int end = mid + 2;

    String ans = str.substring(begin, end);
    return ans;
}
```

[ ]:
```
// https://codingbat.com/prob/p197720

public String left2(String str) {
    String front = str.substring(0, 2);
    String back = str.substring(2);
    return back + front;
```

```
    // return str.substring(2) + str.substring(0, 2);
}
```

[ ]:
```
// https://codingbat.com/prob/p187868

public boolean sleepIn(boolean weekday, boolean vacation) {
    // if (!weekday || vacation) {  // Not good for most new programmers
    if (weekday == false || vacation == true) {
        return true;
    }

    return false;
}
```

[ ]:
```
public int randInt(int min, int max) {
    return (int)(Math.random() * (max - min) + min);
}
```

[ ]:
```
public class Prog88a {
    public static void main(String[] args) {
        /*
        Scanner input = new Scanner(System.in);

        System.out.print("Enter a number between 1 and 13: ");
        int num1 = input.nextInt();
        System.out.println();

        System.out.print("Enter a number between 2 and 20: ");
        int num2 = input.nextInt();
        System.out.println();
        */
        int num1 = (int)(Math.random() * 11 + 10);  // Between 10 to 20
        int num2 = (int)(Math.random() * 21 + 20);  // Between 20 and 40

        int s = num1 + num2;
        int dif = num1 - num2;
        int p = num1 * num2;
        double a = ((double)num1 + (double)num2) / 2;
        int dis = num2 - num1;
        int m = Math.max(num1, num2);

        System.out.println("Num1 = " + num1 + " Num2 = " + num2);
        System.out.println("Sum = " + s);
        System.out.println("Difference = " + dif);
        System.out.println("Product = " + s);
        System.out.println("Average = " + a);
        System.out.println("Distance = " + dis);
```

```
        System.out.println("Maximum = " + m);
    }
}

// Enter a number between 1 and 13: 13
// Enter a number between 2 and 20: 20
// Sum = 33
// Difference = -7
// Product = 33
// Average = 16.5
// Distance = 7
// Maximum = 20
```

1.3 Unit 3 - Boolean Expressions and if Statements

[ ]:
```
public class gradecalc {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        int a = 3; // Ask kids what type a is, then ask what type input is

        System.out.print("Please enter the test score: ");
        int sc = input.nextInt();
        System.out.println();

        String grade = "";

        if (sc >= 90) {
            grade = "A";
        }

        if ((sc >= 80) && (score < 90)) {
            grade = "B";
        }

        if ((sc >= 70) && (score < 80)) {
            grade = "C";
        }

        if ((sc >= 60) && (score < 70)) {
            grade = "D";
        }
        /*
        Kids will try to do
        else {
            grade = "F";
        }
```

```
        However, all the code above is irrelevant; any grade that is not a D␣
 ↪becomes F
        */

        if (sc < 60) {
            grade = "F";
        }

        System.out.println(grade);
    }
}
```

Truth tables: needed for 1 or 2 questions on the AP exam

`if (A or b) and (!A and B)`

A	B	!A	A or B	(!A and B)
T	T	F	T	F
T	F	F	T	F
F	***T***	T	T	***T***
F	F	T	F	F

1.4 Unit 4 - Iteration

[ ]:
```
int counter = 0;

// The opposite of < is >= because of De Morgan's Law
while (counter < 10) {
    System.out.print(counter + " ");
    counter++;  // Modify the Loop Control Variable to prevent infinite loops
}
```

```
0 1 2 3 4 5 6 7 8 9
```

[ ]:
```
int a = 3;
int b = 7;
int c = a + b++;
System.out.println(c);  // c = 10

a = 3;
b = 7;
c = a + ++b;
System.out.println(c);  // c = 11
```

```
10
11
```

[ ]:
```java
import java.util.*;

public class avg {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int cnt = 0;
        double tot = 0;

        while (cnt < 10) {
            System.out.print("Please enter a number: ");
            int num = input.nextInt();
            System.out.println();

            while (!((num > 0) && (num % 2 == 0))) {
                System.out.print("Bad number. Please enter a number: ");
                num = input.nextInt();
                System.out.println();
            }

            cnt++;
            // cnt = cnt + 1;
            // cnt += 1;

            // tot = tot + num;
            tot += num;
        }

        double a = tot / cnt;
        System.out.println("Average = " + a);
    }
}
```

[ ]:
```java
import java.math.*;

public class Prog122d {
    public static void main(String[] args) {
        for (int lcv = -12; lcv <= 16; lcv++) {
            int c1 = lcv;
            double sp1 = Math.pow(lcv, 6);
            double sp2 = -3 * Math.pow(lcv, 5);
            double sp3 = -93 * Math.pow(lcv, 4);
            double sp4 = 87 * Math.pow(lcv, 3);
            double sp5 = 1596 * Math.pow(lcv, 2);
            double sp6 = -1380 * lcv;
            double sp7 = -2800;
            double c2 = sp1 + sp2 + sp3 + sp4 + sp5 + sp6 + sp7;
        }
```

```
    }
}
```

[ ]:
```
public class Prog122h {
    public static void main(String[] args) {
        int i1 = 0;
        double i2 = 0;
        double i3 = 0;
        double i4 = 0;
        double i5 = 0;

        System.out.println("Number\t\tSquare\t\tSquare Root\t\tCube\t\t4th␣
 ↪Root");
        while (i1 < 20) {
            i1++;
            i2 = Math.pow(i1, 2);
            i3 = Math.sqrt((double)i1);
            i4 = Math.pow(i1, 3);
            i5 = Math.pow(i1, 1.0/4);

            System.out.printf(i1 + "\t\t" + Math.round(i2) + "\t\t" + "%.4f",␣
 ↪i3);
            System.out.printf("\t\t\t" + Math.round(i4) + "\t\t" + "%.4f", i5);
            System.out.println();
        }
    }
}
```

[ ]:
```
public class Prog122i {
    public static void main(String[] args) {
        int i1 = -26;
        double i2 = 0;
        double i3 = 0;

        System.out.println("Number\t\tCube Root\t\tCube");
        while (i1 < 25) {
            i1++;
            i2 = Math.pow(i1, 3);
            i3 = Math.cbrt(i1);

            System.out.printf(i1 + "\t\t%.5f\t\t" + Math.round(i2), i3);
            System.out.println();
        }
    }
}
```

prog285b.dat
101 17 2250.00

103 5 4000.00
117 3 7350
118 8 7350.00
125 5 6500.00
138 17 6375
192 8 8125.00
203 8 3250.00
218 5 5000.00
235 5 5250.00
264 17 4150.00
291 17 750.00

[ ]:
```java
/* Template for reading in files */
import java.util.*;
import java.io.*;

public class dfile {
    public static void main(String[] args) {
        try {
            Scanner input = new Scanner(new File("prog285b.dat"));

        } catch (IOException e) {
            System.out.println("Cannot find data file");
        }
    }
}
```

Good supplemental assignments:
58t
82a
93a

2 Day 2 (Units 5-6)

- Modulus - Remainder of long (integer) division
  - Small MOD big = small; 5 % 10 = 5
- As kids get worse at math, section off function quizzes and do one section per day
  - When grading the function quizzes, if kids are getting an average of 7/9, put in gradebook as 7/7 and the people that get 8-9/9 get extra credit
- **Lang285b works great for teaching any new concept**, also great for reviewing all of Units 1-4
  - Try once without arrays, with arrays, then with ArrayLists
- Be careful not to put a line-break at the end of the langdat files, can sometimes mess up Scanner
- If a school offers programming as a math credit, kids are likely to stop taking actual math classes and become likely to fail college math if they go for CompSci especially
- Modify many old programs to incorporate classes
- A class is anything you can think of

- Unit 5 is one of the biggest units in the course, lots of terms to know
- Variable scope appears frequently throughout the AP curriculum
  - Using "my" as a prefix for private class variables helps kids differentiate instance variables
- "The job of the constructor is to set all private data"
- Don't hardcode data **almost ever** – will lose points on the exam if variables were meant to be used
- Show using class calculation methods inside and outside of the class constructor so kids see it in the "Calculation Section" and in the constructor (e.g., Rectangle.getarea())
- Getka helps recruit by being as known in the community as possible and being close to the entrance where kids have to pass by, being the lego robotics coordinator in the district, going to high-level math classes as class selection starts – tell them to stay in upper-level math classes AND take CompSci; recruiting girls takes individual differentiation and strong encouragement that they would succeed
  - Give opportunities for "Academic fairs" – have a (female, ideally) student present lang505t (farm)
  - If a kid finishes AP CSA and can't fit AP 2/CSP into their schedule but they can take Programming 1/2 or some other programming class, let them join that class but study the other class independently
- "The job of the constructor is to set **all private data**"
- Nothing on the exam is case-sensitive unless you use two variables with the same name but different cases
- There is ALWAYS a question on the exam on writing a class and it's the highest-scoring question on the exam – it's almost always the easiest question, so it's very important to get all 9 points on it
  - Points are taken off if the classname is different than they specify on the exam
  - Another point is setting the private class variables, another point is the constructor, and another point is some "get" method
  - A "set" method is not on the exam
- Go back and talk about strings again after learning classes
  - `charAt()`, `toLowerCase()`, `toUpperCase()`, `trim()`, `equalsIgnoreCase()`, `split()` are all very helpful for the exam
  - Students need to know `substring()`, `length()`, `indexOf()`, `equals()`, `compareTo()` for the exam – `equals()` is on **every** exam
- Two string literals with the exact same characters will have the same address – using `new String("...")` will assign it a different address
  - The `==` operator compares object memory addresses, not the contents of the object – use `equals()` instead **always**, otherwise they will lose points on the exam
  - Checking if two strings are in the same memory address isn't on the exam
- Get kids to check out the programming language documentation – if you're the only CS teacher in the school, it's the only way they'll become better than the scope of the class
- In the syllabus, kids must do 20+ hours of programming inside the classroom
- CollegeBoard labs breakout activity:
  - Describe the lab
  - What are the primary concepts that are covered in the lab?
  - What skills must the student have prior to starting the lab?
  - Where would the lab belong in the Course and Exam Description (CED)?
  - Are there any required skills outside the AP Java skillset?
- CollegeBoard labs are great for use after the exam is done

  - Integrate labs **during** the curriculum if possible
- Abstract classes are not on the exam
- Some people like to teach with a lot of pictures/UI versus a lot of math – need to decide on teaching Java Swing/FX or not
- Length of arrays on exam is undetermined – need to know how to use the *length* property
  - Using `arr.length()` instead of `arr.length` is an ungraded error
- Good programmers program like they link
- Stress correct order of arguments in class instantiation
- Give the first AP exam question in class 1-2 days before Christmas break (an easy one like "Cookie Orders")
- 4 hand-written exam questions, #1 is almost always the array question
  - When doing practice questions, show the answer at the end of class even before grading and discuss
- Start writing classes by October

CollegeBoard "Elevens" Lab (2014): * Describe the lab * A Java UI implementation of the Elevens card game; objective of the game is to use all of the cards in the deck to create pairs that add up to 11 * What are the primary concepts that are covered in the lab? * Object-oriented programming, arrays, lists * What skills must the student have prior to starting the lab? * Inheritance, OOP, random, conditionals, loops, arrays, arraylists * Where would the lab belong in the CED? * Most things could be done after Unit 7, everything with inheritance would have to come during/after Unit 9 * Are there any required skills outside the AP Java skillset? * Assertions, Java GUI

[ ]:
```java
public class funcquiz4 {
    public static void main(String [] args) {
        int a=3;
        int b=4;
        int c=5;
        a=4 + a;
        b=c/3 + b/2 +a;
        c=a*a%b%1;
        System.out.println(a+" "+b+" "+c);

        a=3;
        b=4;
        c=5;
        b=a+2/b;
        a=c/3 +b+2 /a;
        c=a/4%a;
        System.out.println(a+" "+b+" "+c);

        a=3;
        b=4;
        c=5;
        a=Math.abs(a-b)/2;
        c=Math.min(a,b) + c/2;
        b=Math.max(Math.max(b+2,3 + a),2);
        System.out.println(a+" "+b+" "+c);
```

```
    }
}

funcquiz4.main(null);
```

```
7 10 0
4 3 1
0 6 2
```

[ ]:
```
import java.util.*;
import java.io.*;

public class Prog285b {
    public static void main(String[] args) {
        try {
            Scanner input = new Scanner(new File("prog285b.dat"));

            while (input.hasNext()) {
                int id = input.nextInt();
                int c = input.nextInt();
                double s = input.nextDouble();
                double com = 0;
                // Rookie students will write "if (c == 5 || 8)" -- doesn't work
                if ((c == 5) || (c == 8)) {
                    if (s <= 5000) {
                        com = s * 0.075;
                    } else {  // Otherwise
                        com = 5000 * 0.075 + (s - 5000) * 0.085;
                    }
                }

                if (c == 17) {
                    if (s <= 3500) {
                        com = s * 0.095;
                    } else {
                        com = 3500 * 0.095 + (s - 3500) * 0.12;
                    }
                }

                System.out.println(id + "   " + c + "   " + s + "   " + com);
            }
        } catch (IOException e) {
            System.out.println("Cannot find data file");
        }
    }
}
```

2.1 Unit 5 - Writing Classes

[ ]:
```
public class Rectangle {
    private int mylen;
    private int mywid;
    private int myarea;
    private int myperm;

    public Rectangle(int l, int w) {
        mylen = l;
        mywid = w;
        myarea = 0;
        myperm = 0;
    }

    public void setarea() {
        // A hard error to find: "int myarea = mylen * mywid;"
        myarea = mylen * mywid;
    }

    public void setperm() {
        myperm = mylen * 2 + mywid * 2;
    }

    public int getarea() {
        return myarea;
    }

    public int getperm() {
        return myperm;
    }
}
```

[ ]:
```
import java.util.*;

// Full Name
// Lang52a
// 6/27/2022

public class Prog52aCL {
    public static void main(String[] args) {
        // Input Section
        Scanner input = new Scanner(System.in);

        System.out.print("What is the length of the rectangle: ");
        int len = input.nextInt();
        System.out.println();
```

```
        System.out.print("What is the width of the rectangle: ");
        int wid = input.nextInt();
        System.out.println();

        // Calculation Section
        Rectangle wow = new Rectangle(len, wid);
        wow.setarea();  // Don't forget to call these
        wow.setperm();

        // Kids are likely to try "getarea()" or "Rectangle.getarea()"
        int area = wow.getarea();
        int perm = wow.getperm();

        // Output Section
        System.out.println("The area of the rectangle is " + area);
        System.out.println("The perimeter of the rectangle is " + perm);
    }
}
```

[ ]:
```
public class salesperson {
    private int myid;
    private int mycode;
    private double mysales;
    private double mycomm;
    public static double PI = 3.14;

    public salesperson(int id, int code, double sales) {
        myid = id;
        mycode = code;
        mysales = sales;
        mycomm = 0;
    }

    public void setcomm() {
        if ((mycode == 5) || (mycode == 8)) {
            if (s <= 5000) {
                mycomm = mysales * 0.075;
                // Kids might try:
                // mycomm = getsales() * 0.075;
                // However, they might also try (which doesn't work):
                // getcomm() = getsales() * 0.075;
            } else {
                mycomm = 5000 * 0.075 + (mysales - 5000) * 0.085;
            }
        }
```

```
        if (mycode == 17) {
            if (s <= 3500) {
                mycomm = mysales * 0.095;
            } else {
                mycomm = 3500 * 0.095 + (mysales - 3500) * 0.12;
            }
        }
    }

    public int getid() {
        return myid;
    }

    public int getcode() {
        return mycode;
    }

    public double getsales() {
        return mysales;
    }

    public double getcomm() {
        return mycomm;
    }

    public String toString() {
        return myid + "    " + mycode + "    " + mysales + "    " + mycomm;
    }
}
```

[ ]:
```
import java.util.*;
import java.io.*;

public class Prog285bCL {
    public static void main(String[] args) {
        try {
            Scanner input = new Scanner(new File("prog285b.dat"));

            while (input.hasNext()) {
                int id = input.nextInt();
                int c = input.nextInt();
                double s = input.nextDouble();

                salesperson wow = new salesperson(id, c, s);

                wow.setcomm();
                System.out.println(wow.toString());
```

```
                // Don't show kids this for a while after doing toString()
                // System.out.println(wow);
            }
        } catch (IOException e) {
            System.out.println("Cannot find data file");
        }
    }
}
```

[ ]:
```
public class stringstuff {
    public static void main(String[] args) {
        String a = "coolbeans";
        System.out.println(a.length());
        System.out.println(a.substring(4, 6));
        System.out.println(a.indexOf("n"));
        System.out.println(a.indexOf("o"));

        String b = "cool";
        // String c = "cool";
        String c = new String("cool");

        System.out.println(a == b);
        System.out.println(c == b);
        System.out.println(c.equals(b));
        System.out.println(c.equals("cool"));
        System.out.println(c.compareTo("cool"));

        for (int lcv = 0; lcv < a.length(); lcv++) {
            // Kids will often do "a.substring(lcv, lcv++)" which is bad
            String spot = a.substring(lcv, lcv + 1);
            // Better to not use charAt because chars aren't on exam
            System.out.println(spot);
        }
    }
}
```

2.2 Unit 6 - Array

[ ]:
```
public class tryit {
    public static void main(String[] args) {
        int[] list = new int[100];

        for (int lcv = 0; lcv < list.length; lcv++) {
            list[lcv] = (int)(Math.random() * 101 - 50);
        }

        for (int x : list) {
```

```
            System.out.println(x);
        }

        int max = list[0];
        for (int lcv = 0; lcv < list.length; lcv++) {
            // max = Math.max(max, a);
            int a = list[lcv];
            if (a > max) {
                max = a;
            }
        }

        System.out.println("The biggest number is " + max);
    }
}
```

[ ]:
```
import java.util.*;
import java.io.*;

public class Prog285bArr {
    public static void main(String[] args) {
        try {
            Scanner input = new Scanner(new File("prog285b.dat"));

            salesperson[] list = new salesperson[1000];
            int cnt = 0;

            while (input.hasNext()) {
                int id = input.nextInt();
                int c = input.nextInt();
                double sales = input.nextDouble();

                salesperson x = new salesperson(id, c, sales);

                // Kids will often try to do "list = x;"
                /* Another common error - overwriting the array:
                salesperson[] wowsa = new salesperson[50];
                list = wowsa;
                */

                list[cnt] = x;
                cnt++;
                // Can also do list[cnt++] = x; but this may be read wrong by␣
 ↪an AP reader
            }
```

```
            // Don't use a for each loop or "lcv < list;" or it will loop 1000␣
 ↪times with null pointers
            for (int lcv = 0; lcv < cnt; lcv++) {
                salesperson z = list[lcv];
                z.setcomm();
            }

            for (int lcv = 0; lcv < cnt; lcv++) {
                salesperson t = list[lcv];
                System.out.println(t.toString());
            }

        } catch (IOException e) {
            System.out.println("Cannot find data file");
        }
    }
}
```

[ ]:
```
public class NumberCube  {
    /** @return an integer value between 1 and 6, inclusive */
    public int toss() {
        return (int)(Math.random() * 6) + 1;
    }
}

public class NumberCubeMain {

    /** Returns an array of the values obtained by tossing a number cube␣
 ↪numTosses times.
     *  @param cube a NumberCube
     *  @param numTosses the number of tosses to be recorded
     *         Precondition: numTosses > 0
     *  @return an array of numTosses values
     */
    public static int[] getCubeTosses(NumberCube cube, int numTosses) {
        int[] tosses = new int[numTosses];
        for (int lcv = 0; lcv < numTosses; lcv++) {
            tosses[lcv] = cube.toss();
        }
        return tosses;
    }

    /** Returns the starting index of a longest run of two or more consecutive␣
 ↪repeated values
     *  in the array values.
     *  @param values an array of integer values representing a series of␣
 ↪number cube tosses
```

```
     *         Precondition: values.length > 0
     *  @return the starting index of a run of maximum size;
     *          -1 if there is no run
     */
    public static int getLongestRun(int[] values) {
        int max = 0;
        int maxIndex = -1;
        int curr = 0;
        for (int lcv = 0; lcv < values.length - 1; lcv++) {
            if (values[lcv] == values[lcv + 1]) {
                curr++;
            } else {
                if (curr > max) {
                    max = curr;
                    maxIndex = lcv - curr;
                }
                curr = 0;
            }
        }
        if (curr > max) {
            max = curr;
            maxIndex = values.length - curr;
        }
        return maxIndex;
    }

    public static void main(String[] args) {
        NumberCube x = new NumberCube();
        int[] y = getCubeTosses(x, 18);
        for (int lcv = 0; lcv < y.length; lcv++) {
            System.out.println(y[lcv]);
        }
        System.out.println("Longest run: " + getLongestRun(y));
    }
}
```

3 Day 3 (Units 7-8)

- Free Responses for 2019-2020 AP Test
- Teach `ceil()` right before giving Function Quiz 7
- Hint for class for Lang602b: look at the `salesperson` class structure (4 pieces of private data, 1 was calculated and 3 were read in)
- Do a lot of practice with array and ArrayLists of objects (not just arrays of ints/doubles), they will be on the exam
- Teach arrays before ArrayLists or kids will never use arrays because they're harder
- The "array" question on the exam will be 50/50 either an array or ArrayList
- Don't teach `var` keyword until after the exam

- ArrayList methods on the exam: `size()`, `get()`, `set()`, `add(obj)`, `add(index, obj)`, `remove()`
  - Using `size()` without () is an ungraded error
  - Not useful to know: `isEmpty()`, `clear()`
  - Useful: `indexOf()`
  - Automatic -1 point if kids try to use `.get()` on an array or `[index]` on an ArrayList
- Kids may struggle with how to swap two elements using `set()`, teach them to swap using 3 lines (temp = a, a = b, b = temp)
- AP CSA Java Quick Reference
  - Give kids the reference in the last month before the exam
- When finding biggest/smallest value in array, preset either to **[0]** OR use `Integer.MIN_VALUE` (or MAX_VALUE) – one exam set a null pointer at slot 0
- Accessing each element in an array properly with for loops is 2 points
- Never let kids hard-code in accessing list length as a number, always use `.length`
- Kids will often print instead of returning, which is -2 points – 1 point off for printing and 1 point off for not returning correctly
  - No possible "negative points" – 1 earned point -3 is 1 point
- If you have an if/else and a return in both spots, chances are it's wrong (lose exam points) – even if you only have one return it may be risky
- Cast int to string: `String temp = "" + 12345;`
  - `System.out.println(1 + 4 + "cool" + 1 + 4);` prints off **5cool14**
  - `System.out.println(1 + 4 + "cool" + (1 + 4));` prints off **5cool5**
- Searching and sorting algorithms will never be asked on the written exam, but it may be helpful to understand their code for the reading portion
  - Know linear/binary search, selection/insertion sort
- How do you make your classes look like your school (e.g., equity)?
  - If you recruit pre-calc/calc students into the class, they will generally boost your overall scores, but how do you bring in the "regular" kids?
  - Hour of Code can help recruit students, recruit girls by going to clubs/orgs
  - How do you make learning opportunities the same for different ethnic/gender group students?
  - Women of color are one of the lowest passing groups of students, need to both recruit more and work to differentiate
- Make sure kids know how to read a matrix – just like reading a book (top-left to bottom-right)

3.0.1 No Penalty

- Extraneous code with no side-effect (e.g., valid precondition check, no-op)
- Spelling/case discrepancies where there is no ambiguity*
- Local variable not declared provided other variables are declared in some part
- `private` or `public` quantifier on a local variable
- Missing `public` qualifier on class or constructor header
- Keyword used as an identifier
- Common mathematical symbols used for operators (x dot div <= >= < > not=)
- [] vs. () vs. <>
- = instead of == and vice versa
- `length/size` confusion for array, `String`, `List`, or `ArrayList`; with or without ()
- Extraneous [] when referencing entire array

- [i,j] instead of [i][j]
- Extraneous size in array declaration, e.g., int[size] nums = new int[size];
- Missing ; where structure clearly conveys intent
- Missing { } where indentation clearly conveys intent
- Missing ( ) on parameter-less method or constructor invocations
- Missing ( ) around `if` or `while` conditions

*Spelling and case discrepancies for identifiers fall under the "No Penalty" category only if the correction can be **unambiguously** inferred from context, for example, "ArayList" instead of "ArrayList." As a counterexample, note that if the code declares "`int G=99, g=0;`" then uses "`while (G < 10)`" instead of "`while (g < 10)`", the context does **not** allow for the reader to assume the use of the lowercase variable.***

[ ]:
```
public class funcquiz7 {
    public static void main(String [] args) {
        int a=3;
        int b=4;
        int c=5;
        a=(int)Math.ceil(3/4);
        b=b+c+c+b/2;
        c=c+c/2+c/3+c/4+c/5;
        System.out.println(a+"     "+b+"    "+c);

        a=3;
        b=4;
        c=5;
        a=(int)Math.floor(3/4);
        b=b+c/(2+(c+b))/2;
        c=c/1+c/2+c/3+c/4+c/5;
        System.out.println(a+"     "+b+"    "+c);

        a=3;
        b=4;
        c=5;
        a=(int)Math.ceil(3/4);
        b=b+c+c+b%2;
        c=c%1+c%2+c%3+c%4+c%5;
        System.out.println(a+"     "+b+"    "+c);
    }
}

funcquiz7.main(null);
```

```
0     16    10
0     4    10
0     14    4
```

prog602b.dat
1825 3.25 0

14063 17.06 1
17185 7.93 1
19111 12.00 2
20045 5.00 1
21352 5.84 0
22841 27.9 2
23051 1.55 2
29118 15.02 0

[ ]:
```java
public class internetcustomer {
    private int myacc;
    private int mycode;  // Could also use a String with .equals or .compareTo
    private double myhours;
    private double mycharge;

    public internetcustomer(int acc, int code, double hours) {
        myacc = acc;
        mycode = code;
        myhours = hours;
        mycharge = 0;
    }

    public void setcharge() {
        double surcharge = 0;
        if (mycode == 1) {
            surcharge = 50;
        } else if (mycode == 2) {
            surcharge = 150;
        }

        if (myhours >= 15.01) {
            mycharge = 550 + (30 * (myhours - 15));
        } else if (myhours >= 5.01) {
            mycharge = 200 + (50 * (myhours - 5));
        } else {
            mycharge = 200;
        }
    }

    public int getacc() {
        return myacc;
    }

    public int getcode() {
        return mycode;
    }
```

```
    public double gethours() {
        return myhours;
    }

    public double getcharge() {
        return mycharge;
    }

    public String toString() {
        return myacc + "    " + myhours + "    " + mycode + "    " + mycharge;
    }
}
```

[ ]:
```
import java.util.*;
import java.io.*;

public class Prog602b {
    public static void main(String[] args) {
        try {
            Scanner input = new Scanner(new File("prog602b.dat"));

            internetcustomer[] list = new internetcustomer[1000];
            int cnt = 0;

            while (input.hasNext()) {
                int id = input.nextInt();
                double h = input.nextDouble();
                int c = input.nextInt();
                internetcustomer jill = new internetcustomer(id, c, h);
                list[cnt] = jill;
                cnt++;
            }

            for (int lcv = 0; lcv < cnt; lcv++) {
                // list[lcv].setcharge();
                // Kids will sometimes try to do list.setcharge();
                internetcustomer fred = list[lcv];
                fred.setcharge();
            }

            for (int lcv = 0; lcv < cnt; lcv++) {
                internetcustomer rick = list[lcv];
                System.out.println(rick);
            }
        } catch (IOException e) {
            System.out.println("Cannot find data file");
        }
```

```
    }
}
```

3.1 Unit 7 - ArrayList

[ ]:
```
import java.util.*;
import java.io.*;

public class Prog285bAL {
    public static void main(String[] args) {
        try {
            Scanner input = new Scanner(new File("prog285b.dat"));

            ArrayList<salesperson> list = new ArrayList<salesperson>();

            while (input.hasNext()) {
                int id = input.nextInt();
                int c = input.nextInt();
                double s = input.nextDouble();

                salesperson wow = new salesperson(id, c, s);
                list.add(wow);
            }

            for (int lcv = 0; lcv < list.size(); lcv++) {
                salesperson mike = list.get(lcv);
                mike.setcomm();
            }

            for (salesperson x : list) {
                System.out.println(x);
            }
        } catch (IOException e) {
            System.out.println("Cannot find data file");
        }
    }
}
```

[ ]:
```
import java.util.*;

public class bigArrayList {
    public static void main(String[] args) {
        ArrayList<Integer> ALlist = new ArrayList<Integer>();

        int[] list = new int[19];
        for (int lcv = 0; lcv < 19; lcv++) {
            int num = (int)(Math.random() * 71 + 20);
```

```
            list[lcv] = num;
            ALlist.add(num);
        }

        // 1
        for (int lcv = 0; lcv < 19; lcv++) {
            System.out.println(list[lcv] + "    " + ALlist.get(lcv));
        }

        // 2
        System.out.println("Array");
        for (int x : list) {
            System.out.println(x);
        }

        System.out.println("ArrayList");
        for (int y : ALlist) {
            System.out.println(y);
        }

        // 3
        System.out.println("Middle");
        System.out.println(list[list.length/2]);
        System.out.println(ALlist.get(ALlist.size()/2));

        // 4
        System.out.println("Average");
        double ans1 = (list[0] + list[list.length - 1] + list[list.length / 2])␣
↪/ 3.0;
        double ans2 = (ALlist.get(0) + ALlist.get(ALlist.size() - 1) + ALlist.
↪get(ALlist.size() / 2)) / 3.0;

        // 5
        int sm = list[0];
        int smslot = 0;
        for (int lcv = 1; lcv < list.length; lcv++) {
            if (list[lcv] < sm) {
                sm = list[lcv];
                smslot = lcv;
            }
        }
        System.out.println("Small is " + sm);

        int big = Integer.MIN_VALUE;
        int bigslot = 0;
        for (int lcv = 0; lcv < ALlist.size(); lcv++) {
            if (ALlist.get(lcv) > big) {
```

```
                big = ALlist.get(lcv);
                bigslot = lcv;
            }
        }

        /*
        big = Integer.MIN_VALUE;
        for (int n : ALlist) {
            if (n > big) {
                big = n;
            }
        }
        */
        System.out.println("Big is " + big);

        // 6
        int temp = list[smslot];
        list[smslot] = list[bigslot];
        list[bigslot] = temp;

        int temp2 = ALlist.get(smslot);
        ALlist.set(smslot, ALlist.get(bigslot));
        ALlist.set(bigslot, temp2);

        // 7
        int nn = (int)(Math.random() * 10);
        ALlist.add(ALlist.size() / 2, nn);

        int[] wowsa = new int[20];
        // Copy top half of list
        for (int lcv = 0; lcv < list.length / 2; lcv++) {
            wowsa[lcv] = list[lcv];
        }
        // Copy bottom half of list
        for (int lcv = list.length / 2; lcv < list.length; lcv++) {
            wowsa[lcv + 1] = list[lcv];
        }
        wowsa[wowsa.length / 2 - 1] = nn;
        list = wowsa;

        System.out.println("New list with nn = " + nn);
        for (int lcv = 0; lcv < list.length; lcv++) {
            System.out.println(list[lcv]);
        }
    }
}
```

[ ]:
```
public class Digits {
    /** The list of digits from the number used to construct this object.
     *  The digits appear in the list in the same order in which they appear in␣
 ↪the original number.
     */
    private ArrayList<Integer> digitList;

    /** Constructs a Digits object that represents num.
     *  Precondition: num >= 0
     */
    public Digits(int num) {
        digitList = new ArrayList<Integer>();
        while (num > 0) {
            digitList.add(0, num % 10);
            num /= 10;
        }
        if (digitList.size() == 0) {
            digitList.add(0);
        }
    }

    /** Returns true if the digits in this Digits object are in strictly␣
 ↪increasing order;
     *  false otherwise.
     */
    public boolean isStrictlyIncreasing() {
        for (int k = 1; k < digitList.size() - 1; k++) {
            if (digitsList.get(i) >= digitsList.get(i + 1)) {
                return false;
            }
        }
        return true;
    }
}
```

prog505a.dat
Sam Summer 4
Linda Lazy 2
Paul Prodder 5
K.C. Master 8
Richie Reader 6

[ ]:
```
public class books {
    private String myName;
    private int myBooks;
    private int myPoints;
```

```
    public books(String name, int books) {
        myName = name;
        myBooks = books;
        myPoints = 0;
    }

    public void calc() {
        if (myBooks <= 3) {
            myPoints = myBooks * 10;
        } else if (myBooks <= 6) {
            myPoints = (myBooks - 3) * 15 + 30;
        } else {
            myPoints = (myBooks - 6) * 20 + 30 + 45;
        }
    }

    public String getName() {
        return myName;
    }

    public int getBooks() {
        return myBooks;
    }

    public int getPoints() {
        return myPoints;
    }

    public String toString() {
        return myName + "  " + myBooks + "    " + myPoints;
    }
}
```

[ ]:
```
import java.util.*;
import java.io.*;

public class Prog505a {
    public static void main(String[] args) {
        try {
            Scanner input = new Scanner(new File("prog505a.dat"));

            List<books> list = new ArrayList<books>();
            while (input.hasNext()) {
                String fname = input.next();
                String lname = input.next();
                int books = input.nextInt();
                books amy = new books(fname + " " + lname, books);
```

```
                list.add(amy);
            }

            for (int lcv = 0; lcv < list.size(); lcv++) {
                books wowsa = list.get(lcv);
                wowsa.calc();
            }

            for (books x : list) {
                System.out.println(x);
            }

            double tot = 0;
            for (books b : list) {
                tot += b.getPoints();
            }
            double avg = tot / list.size();

            int bigpoints = list.get(0).getPoints();
            String bigname = list.get(0).getName();

            for (int lcv = 0; lcv < list.size(); lcv++) {
                books peop = list.get(lcv);
                if (peop.getPoints() > bigpoints) {
                    bigpoints = peop.getPoints();
                    bigname = peop.getName();
                }
            }

            System.out.println("Average points: " + avg);
            System.out.println("The winner is " + bigname);

        } catch (IOException e) {
            System.out.println("Cannot find data file");
        }
    }
}
```

3.2 Unit 8 - 2D Array

prog464a.dat

45 67 89 12 -3
-3 -6 -7 -4 -9
96 81 -8 52 12
14 -7 72 29 -1
19 43 28 63 87

[ ]:
```java
import java.util.*;
import java.io.*;

public class Prog470c {
    public static void main(String[] args) {
        try {
            Scanner input = new Scanner(new File("prog464a.dat"));

            int[][] mat = new int[5][5];

            for (int row = 0; row < mat.length - 1; row++) {
                for (int col = 0; col < mat[0].length - 1; col++) {
                    mat[row][col] = input.nextInt();
                }
            }

            for (int row = 0; row < mat.length - 1; row++) {
                for (int col = 0; col < mat[0].length - 1; col++) {
                    System.out.print(mat[row][col] + "\t");
                }
                System.out.println();
            }

            System.out.println();

            for (int row = 0; row < mat.length - 1; row++) {
                for (int col = 0; col < mat[0].length - 1; col++) {
                    mat[row][5] += mat[row][col];
                    mat[5][col] += mat[row][col];
                    mat[5][5] += mat[row][col];
                }
            }

            for (int row = 0; row < mat.length; row++) {
                for (int col = 0; col < mat[0].length; col++) {
                    System.out.print(mat[row][col] + "\t");
                }
                System.out.println();
            }


        } catch (IOException e) {
            System.out.println("Cannot find data file");
        }
    }
}
```

[ ]:
```java
import java.util.*;
import java.io.*;

public class Prog464a {
    public static void main(String[] args) {
        try {
            Scanner input = new Scanner(new File("prog464a.dat"));

            int[][] mat = new int[5][6];

            for (int row = 0; row < mat.length; row++) {
                for (int col = 0; col < mat[0].length - 1; col++) {
                    mat[row][col] = input.nextInt();
                }
            }

            for (int row = 0; row < mat.length; row++) {
                int rowbig = Integer.MIN_VALUE;
                for (int col = 0; col < mat[0].length - 1; col++) {
                    if (mat[row][col] > rowbig) {
                        rowbig = mat[row][col];
                    }
                }
                mat[row][5] = rowbig;
            }

            for (int[] row : mat) {
                for (int num : row) {
                    System.out.print(num + " ");
                }
                System.out.println();
            }
        } catch (IOException e) {
            System.out.println("Cannot find data file");
        }
    }
}
```

[ ]:
```java
public class GrayImage {
    public static final int BLACK = 0;
    public static final int WHITE = 255;

    /** The 2-dimensional representation of this image. Guaranteed not to be␣
 ↪null.
    *   All values in the array are within the range [BLACK, WHITE], inclusive.
    */
    private int[][] pixelValues;
```

```
    /** @return the total number of white pixels in this image.
    *   Postcondition: this image has not been changed.
    */
    public int countWhitePixels() {
        int count = 0;
        for (int r = 0; r < pixelValues.length; r++) {
            for (int c = 0; c < pixelValues[0].length; c++) {
                if (pixelValues[r][c] == WHITE) {
                    count++;
                }
            }
        }
        return count;
    }

    /** Processes this image in row-major order and decreases the value of each␣
 ↪pixel at
    *   position (row, col) by the value of the pixel at position (row + 2, col␣
 ↪+ 2) if it exists.
    *   Resulting values that would be less than BLACK are replaced by BLACK.
    *   Pixels for which there is no pixel at position (row + 2, col + 2) are␣
 ↪unchanged.
    */
    public void processImage() {
        for (int r = 0; r < pixelValues.length - 2; r++) {
            for (int c = 0; c < pixelValues[0].length - 2; c++) {
                pixelValues[r][c] = Math.max(pixelValues[r][c] - pixelValues[r␣
 ↪+ 2][c + 2], BLACK);
            }
        }
    }
}
```

4 Day 4 (Units 9-10)

- Review previous units using lang480a
- Use function quizzes as a class warmup
  - If testing one section out of ten, Getka puts the quiz in the gradebook out of 10 minus the amount of points they lost
- https://myap.collegeboard.org
  - AP Classroom progress checks are intended to be used as a formative assessment
- Barron's book is the closest book to the AP exam
  - Over winter break, assign a few questions; over spring break, assign a full mock exam with only 90 minutes allowed and have students note where they left off after 90 minutes
- Train your kids to know that the exam is not like other tests that they can probably pass very easily; encourage them never to give up

- Hierarchy for inheritance: "Has a -" for a class containing a property, "Is a -" for a class that inherits from another class
- Overloading: changing a function of the same name in the same class; Overriding: change a function of the same name in a subclass or superclass
- Formal parameters: parameters in the class; Actual parameters: parameters in the function
- Java does not support multiple inheritance
- `super()` call is not necessary to call if no arguments are being passed because it's the default constructor (don't teach this early)
- Interfaces are removed from the AP subset
- Kids need to know how to do something up to a POINT, not just to the end (like reading a file up to a certain marking point, rather than until the end of the file)
- Computer Science as a science credit instead of math?
- AP Classroom has written-response questions, quizzes can be generated on paper and auto-graded
- https://classroom.github.com/
  - Pressing "." in a GitHub repository will open VS code in the browser, update and commit to save file online
  - GitHub repositories will let you check the code updates/commits and see if it was written by another student
  - GitHub classroom has an autograder, also can make a *starter repository* with starting files and/or instruction files/README
  - Replit has a very similar system with the addition of group programming projects
- VS Code
  - Code Runner Extension
  - Java Extension Pack
  - Jupyter Notebooks
  - VS Code also has a git extension and GitHub classroom extension, as well as ssh support
  - VS Code has a file timeline which allows you to see the file history and compare versions
- Writing a Programming Language Handbook using Jupyter Notebooks
- AP 2 - mostly competition programs until beginning of November
- Have to be a teacher at your school for at least 3 years to be an AP grader; should do it at least once
  - ~50-70 readers per question, each assigned a co-reader, 3-4 sets of people assigned as table leader, 3-4 sets of people in one room
- Show **bubble, insertion, and selection sorts** all at the same time
  - *Bubble* the biggest number to the top
  - *Insert* the numbers where they belong
  - *Select* the smallest number first
  - Selection sort MAY have multiple choice questions on the exam, both other sorts would never need to be implemented
  - Show kids the visualizations; let them notice that these three are the slowest ones but don't tell them until they see
- https://www.toptal.com/developers/sorting-algorithms
- https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html
- Almost **always** the "Mystery" problem on the exam is recursion
  - Draw a box to show the recursion stack
  - Recursion isn't a tested *writing* skill on the exam, just reading
  - For recursion practice, codingbat has many

- **Prog702p:** how do you calculate the most common letters in the two words?
  - Use an int array of length 26 to count the frequency of each letter, search for largest one
  - Using a dictionary or map, count the frequency of each letter
  - Use an ArrayList to hold the most frequent letters; use a private variable to keep track of the most frequent letter and its frequency, loop compare with each letter and if a new letter is bigger or equal then add it to the ArrayList, etc.

4.0.1 Exam Prep

- Section 1 - Multiple Choice
  - 40 questions
  - 90 minutes
  - 50% of the exam score
- Section 2 - Free Response
  - 4 questions
    - ArrayList manipulation
    - Class design
    - String analysis & manipulation
    - 2D array traversal and manipulation
  - 90 minutes
  - 50% of the exam score

4.0.2 Great Coding Resources

- CodingBat
- Chortle Java Textbook - good free textbook with quizzes and exercises as well
- Code in the Browser - Labs for teaching the essential qualities of computers using little snippets of code in the browser
- Nifty - is all about gathering and distributing great assignment ideas and their materials
- Rubber Duck Debugging
- Codecademy
- Project Euler
- IntroComputing

4.0.3 Online IDEs

- CompileJava
- Replit

4.0.4 Post AP Ideas/Random Things

- HSCTF - A great after AP Test Activity and free CS competition for your students
- McGov Riddles - Similar to this (not coding, but fun problem solving)
- Souvlaki - RGB Color Exploration / ColorCoder game

[ ]:
```
public class funcquiz9 {
    public static void main(String [] args) {
        int a=3;
        int b=4;
```

```
        int c=5;
        a=a/b    +b/a;
        b=a+b+c/3;
        c+=  c+c;
        System.out.println(a+"     "+b+"    "+c);

        a=3;
        b=4;
        c=5;
        a++;
        b+=b+1;
        c=c/5  + 5/c   + c%5    +  5%c;
        System.out.println(a+"     "+b+"    "+c);

        a=3;
        b=4;
        c=5;
        a=a+b%3   +  c/2  *  2/3;
        c=c+b%1   + 1%b;
        b=c+a     *    4%3   +  3%4;
        System.out.println(a+"     "+b+"    "+c);
    }
}

funcquiz9.main(null);
```

```
1     6    15
4     9    2
5     11    6
```

Bubble Sort	Insertion Sort	Selection Sort
482513	428513	428513
245138	428513	128543
241358	248513	128543
213458	248513	123548
123458	245813	123458
123458	124583	123458
	123458	

[ ]:
```
public int[] selectionSort(int[] list) {
    for (int lcv2 = 0; lcv2 < list.length; lcv2++) {
        int sm = list[lcv2];
        int smSpot = lcv2;
        for (int lcv = lcv2; lcv < list.length; lcv++) {
            if (list[lcv] < sm) {
                sm = list[lcv];
```

```
                smSpot = lcv;
            }
        }
        int temp = list[lcv2];
        list[lcv2] = list[smSpot];
        list[smSpot] = temp;
    }
    return list;
}

int[] nums = {4, 2, 8, 5, 1, 3};
selectionSort(nums);
for (int i = 0; i < nums.length; i++) {
    System.out.print(nums[i] + " ");
}
```

```
1 2 3 4 5 8
```

4.0.5 Practice CED MC Questions - Answer Key

1. **C**
2. **A**
3. **D**
4. **E**
5. **B**
6. **A**
7. **C**
8. **A**
9. **A**
10. **D**
11. **E**
12. **D**
13. **B**
14. **D**
15. **A**

4.1 Unit 9 - Inheritance

prog701g.txt

```
1
Billy
Buckner
3.25
2
Fred
Ballony
28
3
```

Nick
Cuccia
coolbeans
1
Murray
Cox
4.00
1
Carly
Seifert
2.58
1
Elias
Smith
3.22
2
Katy
Rumberger
45
2
Tanya
Barton
78
2
Casey
Bats
97
2
Brandon
Davis
68
3
Ingrid
Sink
superdude
3
Nico
Binge
attaway
3
Mike
Break
done
1
Brad
Williamson
2.75
1

Lorenzo
Rapp
2.55
99

[ ]:
```
public interface names {
    public String getName();
    public String getWord();
}
```

[ ]:
```
public class Animal implements names {
    private String myName;
    private String myWord;

    public Animal() {
        myName = "";
        myWord = "";
    }

    public Animal(String name, String word) {
        // Swapped order to show kids the order doesn't matter
        myWord = word;
        myName = name;
    }

    public String getWord() {
        return myWord;
    }

    public String getName() {
        return myName;
    }
}
```

[ ]:
```
public class Hicca extends Animal {
    private double myFur;

    public Hicca(String name, String word, double cost) {
        super(name, word);
        myFur = cost;
    }

    public double getFur() {
        return myFur;
    }
}
```

[ ]:
```
public class Wallie extends Animal {
    public int mySteps;

    public Wallie(String wname, String word, int steps) {
        // Use a different variable name for "name" to show that it doesn't␣
 ↪have to match
        super(wname, word);
        mySteps = steps;
    }

    public int getSteps() {
        return mySteps;
    }
}
```

[ ]:
```
public class Beeper extends Animal {
    public String myExtWord;

    public Beeper(String name, String word, String extWord) {
        super(name, word);
        myExtWord = extWord;
    }

    public String getExtWord() {
        return myExtWord;
    }
}
```

[ ]:
```
import java.util.*;
import java.io.*;

public class Prog702p {
    public static void main(String[] args) {
        try {
            Scanner input = new Scanner(new File("prog701g.txt"));

            ArrayList<Animal> list = new ArrayList<Animal>();

            int c = input.nextInt();
            while (c != 99) {
                String n = input.next();
                String w = input.next();
                if (c == 1) {
                    double cost = input.nextDouble();
                    // Compiles like a...   vs.   Runs like a...
                    Animal wow = new Hicca(n, w, cost);
                    list.add(new Hicca(n, w, cost));
```

```
                // Does not compile
                // System.out.println(wow.getFur());

                // Compiles
                // System.out.println((Hicca)wow.getFur());
                // System.out.println(wow.getName());
            } else if (c == 2) {
                int steps = input.nextInt();
                Animal wallie = new Wallie(n, w, steps);
                list.add(wallie);
            } else if (c == 3) {
                String extWord = input.next();
                Animal beeper = new Beeper(n, w, extWord);
                list.add(beeper);
            }
            c = input.nextInt();
        }

        double tot = 0;
        int cnt = 0;
        for (int lcv = 0; lcv < list.size(); lcv++) {
            Animal fred = list.get(lcv);
            if (fred instanceof Hicca) {
                Hicca bob = (Hicca)fred;
                tot += bob.getFur();
                cnt++;
            }
        }
        System.out.println("The average hicca fur cost is " + (tot / cnt));

        double totsteps = 0;
        int cntw = 0;
        for (int lcv = 0; lcv < list.size(); lcv++) {
            Animal wowsa = list.get(lcv);
            if (wowsa instanceof Wallie) {
                Wallie bob = (Wallie)wowsa;
                totsteps += bob.getSteps();
                cntw++;
            }
        }
        System.out.println("The average steps taken by wallies is " +␣
↪(totsteps / cntw));

        double len = 0;
        double cntb = 0;
        for (int lcv = 0; lcv < list.size(); lcv++) {
```

```
                Animal pill = list.get(lcv);
                if (pill instanceof Beeper) {
                    Beeper cool = (Beeper)pill;
                    String w = cool.getExtWord();
                    int l = w.length();
                    len += l;
                    cntb++;

                    // Also need to teach this for the advanced kids
                    // len += ((Beeper)cool).getExtWord().length();
                }
            }
            System.out.println("The average length of beepers second word is "␣
↪+ (len / cntb));

            // Find the most common letter
            String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
            int[] counts = new int[26];

            String giant = "";
            for (Animal x : list) {
                if (x instanceof Beeper) {
                    Beeper cool = (Beeper)x;
                    giant += cool.getWord() + cool.getExtWord();
                }
            }

            giant = giant.toUpperCase();
            // System.out.println(giant);

            for (int lcv = 0; lcv < giant.length(); lcv++) {
                String a = giant.substring(lcv, lcv + 1);
                int spot = alphabet.indexOf(a);
                counts[spot]++;
            }

            int big = counts[0];
            int bigspot = 0;

            for (int lcv = 0; lcv < counts.length; lcv++) {
                if (counts[lcv] > big) {
                    big = counts[lcv];
                    bigspot = lcv;
                }
            }
```

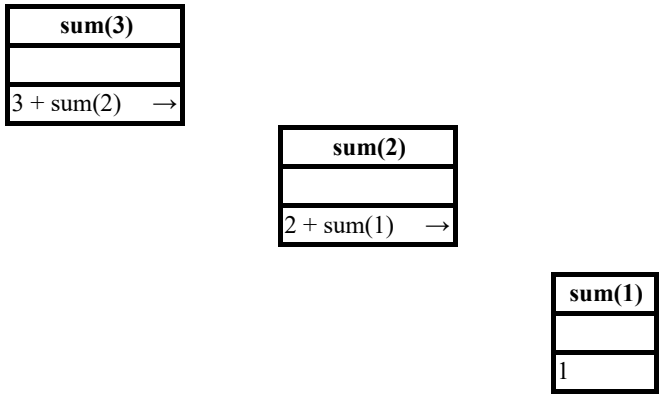
```
            System.out.println("The letter that appears most often is " +␣
 ↪alphabet.substring(bigspot, bigspot + 1));

        } catch (IOException e) {
            System.out.println("Cannot find data file");
        }
    }
}
```

[ ]:
```
public class StepTracker {
    private int minNumSteps;
    private int activedays;
    private int totDays;
    private int totSteps;

    // Could also do
    // private int minSteps;
    // private ArrayList<Integer> dailySteps;

    public StepTracker(int steps) {
        minNumSteps = steps;
        activedays = 0;
        totDays = 0;
        totSteps = 0;
    }

    public void addDailySteps(int steps) {
        if (steps > minNumSteps) {
            activedays++;
        }
        totDays++;
        totSteps += steps;
    }

    public double averageSteps() {
        // Most common point lost for not protecting against zero division
        if (totDays == 0) {
            return 0;
        }
        return (double)totSteps / totDays;
    }

    public int activeDays() {
        return activedays;
    }
}
```

4.2 Unit 10 - Recursion

[ ]:
```
public int sum(int x) {
    if (x <= 1) {
        return 1;
    }
    return x + sum(x - 1);
}

public int box(int x) {
    if (x <= 1) {
        return 1;
    }
    return box(x - 2) + x + box(x - 1);
}
```

4.2.1 Box Method for Recursion




```
6 = 3 + 2 + 1
```

prog702q.txt

1
Bill
4
10000
2
Fred
18
125000
3
Bussy
16
Madison
1
Brian
5
12500

1
Nick
6
11450
2
Billy
17
95000
2
Sally
19
25000
2
Erik
22
101001
2
Softy
19
77000
3
Russ
12
Minneapolis
3
Gus
14
Milwaukee
3
MacBus
13
Oshkosh
3
Babus
14
GreenBay

[ ]:
```
public interface names1 {
    public String getName();
    public int getTire();
    public double getValue();
}
```

[ ]:
```
public class Veh implements names1 {
    private String myName;
    private int myTires;
    private double myValue;
```

```
    public Veh(String name, int tires, double value) {
        myName = name;
        myTires = tires;
        myValue = value;
    }

    public String getName() {
        return myName;
    }

    public int getTire() {
        return myTires;
    }

    public double getValue() {
        return myValue;
    }
}
```

[ ]:
```
public class Car extends Veh {
    public Car(String name, int tires, double value) {
        super(name, tires, value);
    }
}
```

[ ]:
```
public class Truck extends Veh {
    private int myMiles;

    public Truck(String name, int tires, int m) {
        super(name, tires, 50000 - (m * 0.25));
        myMiles = m;
    }

    public int getMiles() {
        return myMiles;
    }
}
```

[ ]:
```
public class Bus extends Veh {
    private String myHome;
    public Bus(String name, int tires, String base) {
        super(name, tires, 50000);
        myHome = base;
    }

    public String getHomebase() {
```

```
        return myHome;
    }
}
```

[ ]:
```
import java.util.*;
import java.io.*;

public class Prog702q {
    public static void main(String[] args) {
        try {
            Scanner input = new Scanner(new File("prog702q.txt"));

            ArrayList<Veh> list = new ArrayList<Veh>();
            while (input.hasNext()) {
                int type = input.nextInt();
                String name = input.next();
                int tires = input.nextInt();
                String value = input.next();
                if (type == 1) {
                    list.add(new Car(name, tires, Double.parseDouble(value)));
                } else if (type == 2) {
                    list.add(new Truck(name, tires, Integer.parseInt(value)));
                } else if (type == 3) {
                    list.add(new Bus(name, tires, value));
                }
            }

            // Calculate the total number of vehicles
            int totVeh = list.size();
            System.out.println("The total number of vehicles is " + totVeh);

            // Calculate the total amount that the cars are worth
            double totCars = 0;
            for (Veh x : list) {
                if (x instanceof Car) {
                    totCars += x.getValue();
                }
            }
            System.out.println("The total amount that the cars are worth is " +␣
 ↪totCars);

            // Calculate the total amount that the vehicles are worth
            double totVehicles = 0;
            for (Veh x : list) {
                totVehicles += x.getValue();
            }
```

```
            System.out.println("The total amount that the vehicles are worth is␣
↪" + totVehicles);

            // Report the longest home destination name for all of the busses
            String longest = "";
            for (int lcv = 0; lcv < list.size(); lcv++) {
                if (list.get(lcv) instanceof Bus) {
                    Bus cool = (Bus)list.get(lcv);
                    if (cool.getHomebase().length() > longest.length()) {
                        longest = cool.getHomebase();
                    }
                }
            }
            System.out.println("The longest home destination name for all of␣
↪the busses is " + longest);

            // Report which truck has the least value
            double least = Double.MAX_VALUE;
            String leastname = "";
            for (Veh x : list) {
                if (x instanceof Truck) {
                    if (x.getValue() < least) {
                        least = x.getValue();
                        leastname = x.getName();
                    }
                }
            }
            System.out.println("The least value truck is " + leastname + " with␣
↪a value of " + least);

            // Report the total number of tires in each of the three classes of␣
↪vehicles
            int totTires = 0;
            for (Veh x : list) {
                totTires += x.getTire();
            }
            System.out.println("The total number of tires in each of the three␣
↪classes of vehicles is " + totTires);

        } catch (IOException e) {
            System.out.println("Cannot find data file");
        }
    }
}
```

Done!