# Course Description

The art and science of computation predates the modern computer by centuries, and in fact is built upon the art of logic going all the way back to authors such as Aristotle. The modern computer is a physical model for repeated uses Boolean logic in the same way that the abacus is a physical model for repeated steps for performing numerical computation.

The goals of this course are to give students (1) a grounding in the progression of logic from Aristotle through Boole, (2) a foundation in basic coding, and (3) the philosophical connection between the two, (4) initiation into seeing the world through the lens of computation, which is proving a rethinking of the world on a scale at least as large as the introduction of calculus caused, and (5) an understanding of the limits of formal logic and computation. All of this is oriented towards the end of getting students to think computationally and algorithmically.

# Institutional Learning Outcomes

# Required Course Materials

**Course Resources**

* Primary Source Reading Packet
* Some teacher resource (online)

TO DO: Go through this file, get course readings, and create a PDF reader.

# Topic Overview

**Table of Contents**

Week 1: Logic 3

Week 2: Logic and a First Program 6

Week 3: Logic and Conditional Programming 9

Week 4: Logic and More Conditional Programming 12

Week 5: Circuits, Logic Gates, and Iteration 15

Week 6: Iteration 17

Week 7: Iteration 19

Week 8: Formal Languages and Regular Expressions 21

Week 9: Regular Expressions Lab 23

Week 10: Methods 25

Week 11: Arrays 28

Week 12: Searching and Complexity 30

Week 13: Sorting and the Beginning of Recursion 32

Week 14: Recursion 34

Week 15: P/NP/NP Complete 37

Week 16: Objects 39

Week 17: Hardware 42

Week 18: Semester Exam 44

Week 19: A Return to Aristotle 46

Week 20: A Return to Aristotle 49

Week 21: Inheritance and Polymorphism 52

Week 22: Polymorphism Lab 54

Week 23: The Parse Tree Logic Lab 56

Week 24: Stacks 58

Week 25: Hash Tables 60

Week 26: Heuristics 62

Week 27: Heuristics Lab 64

Week 28: Turing Machines, Computational Power, and von Neumann Architecture 66

Week 29: The Limits of Computation 69

Weeks 30–34: Final Project 71

Week 35: Man and Technology 73

Week 36: Final Exam 75

|  |  |  |
| --- | --- | --- |
| Week 1: Logic |  | Quarter 1 |
| **Weekly Overview** | |  |
| This week presents and introduction to logic, culminating in a reading and discussion about the limits of computer “understanding.” The idea is to frame the course for what it is: a computer is a physical model of a formal logical system in the same way that an abacus is a physical model of arithmetic. It is also to frame from the beginning that a computer cannot “know.” That is why we start with a distinction between formal and material logic and a discussion about judgments and truth. | |  |
| **Main Objectives** | |  |
| * Understand the distinction between formal and material logic. * Understand the basic framework of logic: conceiving, judging, reasoning, terms, propositions, and arguments. * Understand the limitations of a computer. | | |
| **Required Student Resources** | | |
| **Readings**  *Prior Analytics* I.1, *Posterior Analytics* I.1, I.2  “All of Logic in Two Pages”, from *Socratic Logic,* Peter Kreeft, pages 26-27  “Understanding, the thing that distinguishes man from both beast and computer”, from *Socratic Logic,* Peter Kreeft, pages 25-40 | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1: Course Introduction**  The teacher will want to use most of this period to introduce the course, assign seats, and other general “setup” items for the year. If there is time, begin reading *Prior Analytics* I.1 together.  **Homework:** Read *Prior Analytics I.1* and *Posterior Analytics I.1, I.2*  1 | | |
| **Day 2: Formal and Material Logic Introduction**  Conduct a seminar on the Aristotle readings. See the seminar notes on important points and questions.  **Homework:** Read “All of Logic in Two Pages”, P. Kreeft  TODO: Write seminar questions and discussion notes (1-2 pages) on *Prior Analytics I.1* and *Posterior Analytics* I.1, I.2  2 | | |
| **Day 3: An Introduction to Logic**   1. Using the reading from the night before, define and connect *conceiving* (*understanding*), *judging*, *reasoning*, *terms*, *propositions*, and *arguments*. 2. Define *clarity* (of terms), *truth*, and *validity.*   **Homework:** Exercises on identifying terms, propositions, and arguments.  TODO: Write lecture notes  TODO: Write HW exercises  3 | | |
| **Day 4: Linking Judgment and Truth**   1. Define judgment and link to propositions. 2. Define subject and predicate. 3. Introduce simple and compound sentences as types of propositions. 4. Simple – categorical 5. Compound – hypothetical, disjunctive, conjunctive   Teacher Note: The introduction of disjunctive and conjunctive is an important moment. It introduces *and* and *or*, which sets the stage for everything that is to come.  **Homework:** Read “Understanding, the thing that distinguishes man from both beast and computer” (P.Kreeft).  TODO: Write lecture notes  TODO: Write HW exercises  4 | | |
| **Day 5: The Philosophical Limits of a Computer**  Conduct a seminar on the Peter Kreeft reading.  TODO: Write seminar questions and discussion notes (1-2 pages) on the Kreeft reading.  5 | | |

|  |  |  |
| --- | --- | --- |
| Week 2: Logic and a First Program |  | Quarter 1 |
| **Weekly Overview** | |  |
| This week focuses on the formalization of logic in truth tables and truth trees, but also situates formal logic properly by returning to judgment and categorical propositions. The constant refrain is (1) a computer is a physical model of formal logic, (2) formal logic is incomplete without material logic, and (3) material logic together with formal logic is the way to truth. This week also sees the introduction of the BlueJ IDE and the writing of the first program. | |  |
| **Main Objectives** | |  |
| * Diagram formal propositions with *and, or,* and *not* using truth tables and truth trees. * Understand the limitations of bivalence. * Prove formal equivalencies with truth tables. * Understand the four categorical propositions. * Launch BlueJ with a simple “Hello World” program. | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1: Truth Tables and Truth Trees**  Introduce the mathematical definition of *and, or*, and *not*, together with their truth tables and truth trees. Students should see examples of propositions that have at least three terms (*p*, *q* and *r*) and that stack up a few operations. Make the connection back to distributed and undistributed terms, and emphasize that the “precise” mathematical (and therefore computer) definition of these terms does not line up with the way that we always use them in language.  This is also the time to introduce the concept of *bivalence* (that every declarative has exactly one truth value) and its limitations, e.g. for future contingents (“There will be a sea battle tomorrow”).  **Homework:** Construct truth tables and truth trees to exhaust compound propositions.  TODO: Lecture Notes for Teacher  TODO: Homework assignment  6 | | |
| **Day 2: Truth Table Proofs**  Students should see all of the basic properties of *and*, *or*, and *not*, including commutativity, associativity, double negation, and DeMorgan’s Laws. They should see “proof by truth table” and be asked to prove various symbolic propositions using this method.  **Homework:** Exercises for truth table proofs  TODO: Lecture Notes for Teacher  TODO: Homework assignment  7 | | |
| **Day 3: Judgment and Categorical Propositions**  We return to the relationship between truth and judgment.   1. Introduce the four types of categorical propositions. 2. Practice identifying different types of propositions 3. Practice writing propositions   **Homework:** Exercises on identifying and writing propositions.  TODO: Lecture Notes for Teacher  TODO: Homework assignment  8 | | |
| **Day 4: Judgment and Categorical Propositions (continued)**  Students will need more practice and an opportunity to go over the homework.  **Homework:** More exercises on identifying and writing propositions.  TODO: Lecture Notes for Teacher  TODO: Homework assignment  9 | | |
| **Day 5: The IDE and the Compile Process**  Have students launch BlueJ, and give a basic description of the code, compile, assemble, execute process (very high level). Have students write the “Hello World” program, making a big deal about what just happened. This is also the time to discuss the anatomy of a Java Program. We will not yet introduce GitHub, and students will not be turning in code.  **Homework:** None  TODO: Write lecture notes for teacher on presenting BlueJ for the first time  TODO: Student handout on BlueJ and the anatomy of a Java Program  10 | | |

|  |  |  |
| --- | --- | --- |
| Week 3: Logic and Conditional Programming |  | Quarter 1 |
| **Weekly Overview** | |  |
| This week focuses on the Square of Opposition, and obversion, conversion, and contraposition. Now that students have the basics of truth tables under their belts, they are ready for conditional programming. Notice that basic I/O is not instructed. Students will abstract this by seeing the code work. We jump right into the use of conditionals (if, then, else) and the logical structures of and (&&), or (||), and not (!). | |  |
| **Main Objectives** | |  |
| * Understand the Square of Opposition. * Produce the obversion, conversion, and contraposition of propositions. * Explore the use of conditionals and Boolean operators in computer code. | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1: The Square of Opposition**   1. Introduce the Square of Opposition. 2. Practice identifying and writing categorical propositions.   **Homework:** Exercises on the Square of Opposition  TODO: Lecture Notes for Teacher  TODO: Homework assignment  11 | | |
| **Day 2: Obversion, Conversion, and Contraposition**  Introduce the three concepts and give examples. There is a need to introduce this in two ways: form the square of opposition, but also from the mathematical perspective (conditional statements, inverse, converse, and contrapositive, together with their respective truth tables). Make a big deal about the mathematical definition of implication when there is a false premise. Mathematically, this makes the implication “true”, but there is a need to once again abstract truth from validity. The introduction of “if, then” will help both with the coding, but also with hypothetical propositions later.  The mathematical definitions allow for the truth tables for conditional and biconditional. Using truth tables, students can derive *modus ponens* and *modus tollens*.  **Homework:** Exercises on obversion, conversion, and contraposition, with truth table proofs of important properties.  TODO: Lecture Notes for Teacher  TODO: Homework assignment  12 | | |
| **Day 3: Obversion, Conversion, and Contraposition (continued)**  Finish the content on these three topics, answer questions on homework, and give more examples.  **Homework:** More exercises on obversion, conversion, and contraposition, with truth table proofs of important properties  TODO: Lecture Notes for Teacher  TODO: Homework assignment  13 | | |
| **Day 4: Conditional Programming Exploration**  Pass out the “Conditional Programming” Lab. The students will also need the existing code. Part of the exploration will be to decide what the symbol is for “and”, “or”, and “not.” No instruction is necessary – this is an exploration lab.  **Homework:** None  TODO: Exploration Lab (code) for students  TODO: Exploration Lab (activities and questions) for students  14 | | |
| **Day 5: Test**  Teat on logic material covered thus far, including conditionals in Java  TODO: Test?  15 | | |

|  |  |  |
| --- | --- | --- |
| Week 4: Logic and More Conditional Programming |  | Quarter 1 |
| **Weekly Overview** | |  |
| The focus this week is on the introduction of syllogism and a more formal presentation of conditional programming. The introduction of syllogisms is the first real “marriage” between formal and material logic. We take true premises and reason with them by use of the syllogism. The lab last week was an exploration of conditionals in Java code. This week has a formal presentation, but also includes instruction in basic data types and operations. | |  |
| **Main Objectives** | |  |
| * Define a syllogism. * Apply Aristotle’s Six Rules for Validity. * Understand data types and operations. * Understand conditional programming and Boolean operators. | | |
| **Required Student Resources** | | |
| **Readings**  *The Pattern on the Stone*, D. Hillis (Chapter 1) | | |
| **Assignments**  The are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  *Need* | | |
| **Daily Outline** | | |
| **Day 1: Aristotelian Syllogisms**  *Note that this is a two-day lesson.*   1. Define syllogism, major premise, minor premise, and conclusion. 2. Introduce Aristotle’s Six Rules for Validity. 3. Practice identifying the categorical syllogisms, writing syllogisms.   **Homework:** Syllogism exercises  TODO: Lecture Notes for Teacher (for both days)  TODO: Homework assignment  16 | | |
| **Day 2: Aristotelian Syllogisms (continued)**  Continue the work from the previous day.  **Homework:** More syllogism exercises  TODO: Homework assignment  17 | | |
| **Day 3: Data Types, Operations, and Conditional Programming**  We begin to hit heavy coding topics here. The goal is to start to build up the basics so that students can utilize them for more complicated tasks. Today has a lecture on data types, basic operations (including %), equality (including “.equals” versus “==” for String), and conditionals (if, else). This lecture should very briefly cover user I/O, which students will have seen through their exploration labs.  **Homework:** Exercises on conditional programming  TODO: Lecture notes  TODO: HW exercises  18 | | |
| **Day 4: Conditional Programming Exericise**  Pass out the in-class programming exercises for conditional and have students work on them. There will be no code collected, but the teacher can do a “checklist” to see how many they have completed. Students may need reminded of how to set up a file, type code (with the basic class anatomy), compile, and execute the code.  **Homework:** Code by hand any exercises you did not complete in class. Read *The Pattern on the Stone*, D. Hillis (Chapter 1)    TODO: In class programming exercises  19 | | |
| **Day 5**  Flexible | | |

|  |  |  |
| --- | --- | --- |
| Week 5: Circuits, Logic Gates, and Iteration |  | Quarter 1 |
| **Weekly Overview** | |  |
| The focus this week takes a giant step forward in connecting formal logic with the machine of a computer. Students have now had truth tables, and they have also spent quite a bit of time learning that formal logic alone does not convey truth. They are ready to see how a “computing device” might be built. To do this, we use series and parallel circuits to model *and* and *or*, and we take the next step to logic gates. In all of this, we take our lead from W. Daniel Hillis in *The Pattern on the Stone*. | |  |
| **Main Objectives** | |  |
| * Understand that parallel and series circuits are physical models of *and* and *or*. * Apply formal logic and circuits to solve a computational problem. * Introduce iteration. | | |
| **Required Student Resources** | | |
| **Readings**  *The Pattern on the Stone*, D. Hillis (Chapter 2) | | |
| **Assignments**  Lab Assignment: Circuits and Rock, Paper, Scissors  Homework Assignment: Logic Gates Exercises  Lab Assignment: Exploring Iteration | | |
| **Extra Teacher Resources** | | |
| **Readings**  The Hillis readings are sufficient for teacher preparation. For more information on actual logic gates design using transistors, see: <http://hyperphysics.phy-astr.gsu.edu/hbase/Electronic/trangate.html#c1> | | |
| **Daily Outline** | | |
| **Day 1: Rock, Paper, Scissors Wiring Lab**  Students will work in pairs to diagram how one would wire a game of Rock, Paper, Scissors with two sets of inputs (one for each player) and three outputs (Player 1 Win, Player 2 Win, Tie). See the lab notes on hints students will need.  **Homework:** Complete the wiring diagram.  20 | | |
| **Day 2: Rock, Paper, Scissors Wiring Lab (continued)**  Students will actually wire their game using batteries and light bulbs.  **Homework:** Read Hillis, Chapter 2  21 | | |
| **Day 3: Logic Gates**  The lecture today is on logic gates (following the Hillis reading) and disjunctive normal form. Students should be introduced to some basic physics of the logic gates, but they are not responsible for a full knowledge of them.  **Homework:** Draw a logic gate diagram for the Rock Paper Scissors machine.  22 | | |
| **Day 4: Exploring Iteration**  This day is an exploration lab. Students should be given some code with a for loop and asked to change parameters and describe certain behaviors and results. They should then be asked to code certain basic for loops.  **Homework:** None  TODO: Exploration Lab (code) for students  TODO: Lab sheet (activities and questions)  23 | | |
| **Day 5: Test**  Test on logic, logic gates, and some conditional programming  TODO: Test?  24 | | |

|  |  |  |
| --- | --- | --- |
| Week 6: Iteration |  | Quarter 1 |
| **Weekly Overview** | |  |
| This week focuses on the for loop and the while loop, as well as some important/famous mathematical algorithms. Iterative thinking is at least some perquisite for the upcoming material on deterministic finite automata, and is certainly relevant for the much later material on Turing Machines. | |  |
| **Main Objectives** | |  |
| * Write a for loop to accomplish various tasks. * Write a while loop to accomplish various tasks. | | |
| **Required Student Resources** | | |
| **Readings**  Need | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1: The For Loop**  This is a lecture on for loops. Lots of examples need given to students.  **Homework:** Two for loop exercises, both of which will be on an upcoming lab.  TODO: Lecture Notes for teacher  TODO: Homework for students  25 | | |
| **Day 2: Basic For Loop Practice**  This is a short lab day in which students are asked to write some very basic, non-interesting for loops.  **Homework:** None  TODO: Lab sheet for students  26 | | |
| **Day 3: The While Loop**  This is a lecture on while loops. Lots of examples need given to students.  **Homework:** Two while while exercises, both of which will be on an upcoming lab.  TODO: Lecture Notes for teacher  TODO: Homework for students  27 | | |
| **Day 4: Basic While Loop Practice**  This is a short lab day in which students are asked to write some very basic, non-interesting while loops.  **Homework:** None  TODO: Lab sheet for students  28 | | |
| **Day 5**  Flexible | | |

|  |  |  |
| --- | --- | --- |
| Week 7: Iteration |  | Quarter 1 |
| **Weekly Overview** | |  |
| This week has the first major multi-day lab, which also necessitates the introduction of GitHub. | |  |
| **Main Objectives** | |  |
| * Write a for loop to accomplish various tasks. * Write a while loop to accomplish various tasks. | | |
| **Required Student Resources** | | |
| **Readings**  “Formal Languages”, <https://introcs.cs.princeton.edu/java/51language/>  *The Pattern on the Stone*, D.W. Hillis (Chapter 2) | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1: Iteration Lab**  This is a lab day. However, it will need to begin with a discussion of GitHub as the mechanism by which students will turn in code. After that, students can work individually on the lab problems. In fact, this may take the better part of the period. The problems in this lab will include the following:   1. Finding the sum of the odd numbers 1 through 2*n* + 1 2. Checking to see if a String is a palindrome 3. Determining if a year is a leap year 4. Checking to see that a number is prime 5. Calculate the *n*th Fibonacci number 6. Finding the greatest common divisor 7. Finding the least common multiple 8. The Collatz Conjecture   **Homework:** None  TODO: Lab sheet for students  TODO: GitHub reference sheet  29 | | |
| **Day 2: Iteration Lab**  Second day of the Iteration Lab  **Homework:** None  30 | | |
| **Day 3: Iteration Lab**  Second day of the Iteration Lab  **Homework:** None  31 | | |
| **Day 4: Iteration Lab**  Third day of the Iteration Lab  **Homework:** None  32 | | |
| **Day 5: Test**  Test on Iteration  **Homework:** Read the Princeton page on Formal and Regular Languages and Regular Expressions, and Hillis, Chapter 2.  TODO: Test?  33 | | |

|  |  |  |
| --- | --- | --- |
| Week 8: Formal Languages and Regular Expressions |  | Quarter 1 |
| **Weekly Overview** | |  |
| This week focuses on formal languages and regular expressions. This is important for two reasons. First, it is the very basics of how a computer processes information, and second because it provides background for the material on Turing Machines. Students will be introduced to formal and regular languages and to regular expressions. They will go back and forth between English descriptions of a regular language and the regular expression that describes the language. They will also be introduced to deterministic finite automata. | |  |
| **Main Objectives** | |  |
| * Define formal language. * Define regular language and regular expression. * Convert between English descriptions of languages and their regular expression. * Build DFAs that represent regular expressions. * Understand Kleene’s Theorem | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  Homework: Regular Expressions  In-Class Exercises: Regular Expressions  Homework: More Regular Expressions  Homework: DFAs  In-Class Exercises: DFAs  Homework: More DFAs | | |
| **Extra Teacher Resources** | | |
| **Readings**  Formal Languages, Regular Expressions, DFAs, and NFAs: <https://introcs.cs.princeton.edu/java/51language/> | | |
| **Daily Outline** | | |
| **Day 1: Formal Languages and Regular Expressions**  Define *symbol*, *alphabet*, *string*, and *formal language*. Introduce the concept of a regular language and how they solve the specification and recognition problems. We will not go into “generalized REs”. There should be lots of examples of describing “in English” what a regular expression represents, e.g. (a|b)\*aa is “all binary strings that end with aa.” We use a syntax for regular expressions that is a subset of the java.util.regex.Pattern syntax.  **Homework:**Regular expression exercises.  34 | | |
| **Day 2: Regular Expressions (continued)**  More exercises will be needed on this day so that students can practice, and we will briefly introduce the java.util.regex.Pattern class.  **Homework:** More regular expression exercises  35 | | |
| **Day 3: The DFA**  Define a deterministic finite state automata. This lesson presents a bunch of examples where students go back and forth between an English description, an RE, and a DFA.  **Homework:** DFA Exercises  36 | | |
| **Day 4: DFAs, NFAs, and Kleene’s Theorem**  The bulk of this lesson is more practice with DFAs. However, the is a big course “climax” here as well. Briefly introduce students to the NFA, to other various modifications of a DFA, and to Kleene’s Theorem. There is a need to emphasize that the computer, for what it is worth, is powerful enough. We can make it faster, but we cannot make it “better.”  **Homework:** More DFA Exercises  37 | | |
| **Day 5**  Flexible | | |

|  |  |  |
| --- | --- | --- |
| Week 9: Regular Expressions Lab |  | Quarter 1 |
| **Weekly Overview** | |  |
| This week has the first major programming lab for students. It is a “word search” type lab that requires knowledge of regular expressions as well as iteration. Students will also see basic file I/O for the first time. | |  |
| **Main Objectives** | |  |
| * Apply the knowledge of regular expressions, iteration, and conditional programming to solve problems. | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  Lab Assignment: Regular Expressions and English Words | | |
| **Extra Teacher Resources** | | |
| **Readings**  The full Java Pattern class API: <https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html>  The full Java String class API: <https://docs.oracle.com/javase/7/docs/api/java/lang/String.html> | | |
| **Daily Outline** | | |
| **Day 1: Regular Expression Lab**  The day should be an introduction to the Pattern class (https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html) as well as some very basic file input code. The lab will give students a “dictionary” (from Scrabble or some other source) and ask them to use REs to find words with certain properties. We will also include some instruction on String methods, which is a good time to introduce the concept of an API.  **Homework:** None  38 | | |
| **Day 2: Regular Expression Lab**  This week is entirely for the lab.  **Homework:** None  39 | | |
| **Day 3 Regular Expression Lab**  This week is entirely for the lab.  **Homework:** None  40 | | |
| **Day 4: Regular Expression Lab**  This week is entirely for the lab.  **Homework:** None  41 | | |
| **Day 5: Test**  Test on regular expressions and DFAs.  TODO: Test?  42 | | |

|  |  |  |
| --- | --- | --- |
| Week 10: Methods |  | Quarter 2 |
| **Weekly Overview** | |  |
| This week focuses on methods. This is the first abstraction in computer coding, (second if you include variables). Students will learn to write methods both for code reuse, but also for ease of understanding and modular thinking. There will be a return to the regular expression lab that will make use of methods. This is also the end of the first quarter, which nicely culminates in the writing of methods and includes a continued emphasis on regular expressions. | |  |
| **Main Objectives** | |  |
| * Understand the mechanics of writing methods in Java. * Understand how parameters and return types are affected by method calling. * Apply methods to modularize existing code. | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1: Methods**  The lecture should be about methods, how to code them, how to call them, and their effect on field values. Particular attention should be given to the way in which methods do not alter the value of local variables in the code that calls the method.  **Homework:** Methods exercises.  TODO: Lecture notes on Methods  TODO: Homework exercieses  43 | | |
| **Day 2: Method Practice**  Students will work on short exercises, both involving the writing of methods and also the “result” of code that calls methods.  **Homework:** None  TODO: In-class exercises  44 | | |
| **Day 3: A Return to REs**  The lab here is shorter. It should amount to (1) taking one of the problems from the RE lab and “methodizing it” (getData, processData, outputData), etc.  **Homework:** None  TODO: Lab sheet for students  45 | | |
| **Day 4: RE Lab Exercise**  This is a different lab exercises, but it will still involve regular expressions. However, it needs designed in a way that forces two people to work together and agree on a method interface. One student will write one method, a second student another method, but they have to get them to work together after they work on them separately.  Students jointly write the Javadoc. One student implements method M and unit tests for the method. The other starts with an empty implementation of method M, and writes unit tests and code that will call M. Student 1 provides M to student 2, and they jointly get the overall system to work.  TODO: how student 1 gets code to student 2.  TOTO: Lab sheet for students  **Homework:** None  46 | | |
| **Day 5**  Flexible. However, a good challenge question/problem to pose would be to ask students to think about how to use a regular expression to describe the arithmetic expressions of Java. It will highlight two things: first, a possible use of regular expressions (which become very, very important at the level of compiler, etc.), and second, the limitation of regular expressions. It turns out that one *cannot* describe the Java language. Structuring this activity will take some thought, but the philosophical ramifications of this could be powerful.  TODO: Optional Activity | | |

|  |  |  |
| --- | --- | --- |
| Week 11: Arrays |  | Quarter 2 |
| **Weekly Overview** | |  |
| This week focuses on arrays, which necessarily includes an emphasis on iteration and methods. For arrays, students should know how they are stored and indexed and how they are affected when they are parameters in a method. There will be a lab that includes a very famous algorithm, the Sieve of Eratosthenes. | |  |
| **Main Objectives** | |  |
| * Understand how to declare an array, set values, and get values. * Understand how an array is stored and indexed. * Understand how an array acts when its reference is passed to a method. * Use arrays to solve mathematical problems. | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1: Arrays**  The focus of this lecture should be on arrays, how to use them in code, and how they are addressed in memory (why indexing at 0 is much more convenient than indexing at 1). During this lecture, we will also need to bring up the concept of a “reference” versus a primitive and how methods effect arrays.  **Homework:** Some array exercises, e.g., write a method that prints out the largest element in an array of ints. Include here also “what does the code do” exercises so as to address the concept of passing references rather than actual objects.  TODO: Lecture notes  TODO: Homework exercises  47 | | |
| **Day 2: Arrays Lab**  This lab should contain three math algorithms: (1) the *Sieve of Eratosthenes*, (2) taking in a number and producing an array of its factors, and then going backwards, and (3) representing a very large number as an array of digits, and then programming addition and subtraction. All problems need to be written with methods.  **Homework:** None  TODO: Lab sheet for students  48 | | |
| **Day 3: Arrays Lab**  Students should continue working on their arrays lab.  **Homework:** None  49 | | |
| **Day 4: Arrays Lab**  Students should continue working on their arrays lab.  **Homework:** None  50 | | |
| **Day 5: Test**  Test on Methods and Arrays  TODO: Test?  51 | | |

|  |  |  |
| --- | --- | --- |
| Week 12: Searching and Complexity |  | Quarter 2 |
| **Weekly Overview** | |  |
| This week begins a more formal presentation of algorithms by looking at searching algorithms and complexity analysis. Students should be able to code linear and binary search and should be able to conceptually define the order of growth of functions to describe the efficiency of an algorithm. | |  |
| **Main Objectives** | |  |
| * Write linear search. * Write binary search. * Describe the efficiency of algorithms using order of growth. * Classify linear search and binary search in terms of the order of growth. | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  Lab Assignment: Searching Lab  Homework Assignment: Complexity Analysis | | |
| **Extra Teacher Resources** | | |
| **Readings**  Complexity analysis: <https://introcs.cs.princeton.edu/java/41analysis/>  Binary Search: <https://en.wikipedia.org/wiki/Binary_search_algorithm>  Searching: <https://introcs.cs.princeton.edu/java/42sort/> | | |
| **Daily Outline** | | |
| **Day 1: Searching**  Have students work on paper to develop linear search on an array. There is no need to code this one. Students should engage in an exercise of determining the worst-case function, best-case function, and average-case function (starting with a table of values, but then getting to an actual function) for the algorithm. This is the introduction to complexity analysis.  **Homework:** None  52 | | |
| **Day 2: The Binary Search**  Open up with the dictionary demonstrations. (Have the student identify any word in the dictionary, and you get 20 “yes or no” questions to guess their word. Discuss how this was done, and have students work to develop a best, worst, and average case for this algorithm. Name the algorithm at the end.  **Homework:** Code binary search on paper (if not completed in class). Read through the StopWatch.java file.  53 | | |
| **Day 3: Searching Lab**  Give students a timer class and rather large data files, and have them write the two search algorithms and time their results.  **Homework:** Do a regression on the data calculate and develop a time function.  54 | | |
| **Day 4: Complexity Analysis**  This is a lecture on complexity analysis, complete with the formal definition of the order of growth for functions, and loaded with tons of for loops and examples.  **Homework:** Complexity analysis exercises.  55 | | |
| **Day 5**  Flexible | | |

|  |  |  |
| --- | --- | --- |
| Week 13: Sorting and the Beginning of Recursion |  | Quarter 2 |
| **Weekly Overview** | |  |
| This week focuses on sorting algorithms, a continued exploration of complexity analysis, and an introduction to recursion. Students will program three quadratic sorts and perform complexity analysis on them. They will also being a discussion of recursion by connecting it back to recursive mathematical formulas. | |  |
| **Main Objectives** | |  |
| * Code selection sort, bubble sort, and insertion sort. * Performa complexity analysis on the three sorts. * Understand the insertion sort can be used on streaming data, whereas the other two cannot. * Understand recursion, both mathematically and in terms of code. | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  Sorting Exploration Student Activity  Modified Sorts Worksheet  Recursion Worksheet | | |
| **Extra Teacher Resources** | | |
| **Readings**  Recursion: <https://introcs.cs.princeton.edu/java/23recursion/>  Sorting descriptions and visualizers: <https://visualgo.net/bn/sorting?slide=1>  Sorting: <https://introcs.cs.princeton.edu/java/42sort/> | | |
| **Daily Outline** | | |
| **Day 1: Exploring Sorting**  Students should work on paper in pairs to write a sorting algorithm.  **Homework:** None  56 | | |
| **Day 2: Selection Sort and Bubble Sort**  Work as a class to understand the algorithm, and time permitting, code it on the board. Perform complexity analysis on these two algorithms.  **Homework:** Modified Sorts worksheet.  57 | | |
| **Day 3: Modified Bubble Sort and Insertion Sort**  This is an important sort because it is the only one so far that can deal with data that is coming in “live” at the end of the stream. Run the class the same way as yesterday.  **Homework:** None.  58 | | |
| **Day 4: Test**  Test on Searching and Sorting.  TODO: Test?  59 | | |
| **Day 5: Introduction to Recursion**  Students should be introduced to recursion, first from a mathematical perspective (recursive formulas versus explicit formulas), but ultimately as a programming technique. This is a completely different way of thinking, and it will take students a while to get used to it.  **Homework:** Short recursion exercises.  60 | | |

|  |  |  |
| --- | --- | --- |
| Week 14: Recursion |  | Quarter 2 |
| **Weekly Overview** | |  |
| This week is a deep dive into recursion. Students will see examples when the recursive call is in the beginning, the middle, and the end. They will program several examples of algorithms they have already seen (gcd, primes, etc.), and also program Merge Sort. With this later topic, they see their first non-quadratic sort. | |  |
| **Main Objectives** | |  |
| * Apply recursive techniques to solve problems. * Code Merge Sort and perform complexity analysis. * Understand the impact of recursive calls on memory. | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  Recursion: <https://introcs.cs.princeton.edu/java/23recursion/>  Sorting descriptions and visualizers: <https://visualgo.net/bn/sorting?slide=1>  Sorting: <https://introcs.cs.princeton.edu/java/42sort/> | | |
| **Daily Outline** | | |
| **Day 1: Recursion Examples**  The entire day is dedicated to writing recursion examples. See the lecture notes for which ones are important.  **Homework:** Recursion exercises (both “write the following …” and “what does the following do …”)  TODO: Lecture notes for teacher  TODO: Homework exercises  61 | | |
| **Day 2: Recursion Examples (continued)**  Go over the exercises and give students a chance to practice more. Here, we should emphasize the difference between examples that have recursive calls at the beginning, in the middle, and at the end. Discuss with students an informal concept of a Stack, and use this to demonstrate why non-tail recursion can cause memory problems.  **Homework:** More recursion exercises.  TODO: Lecture notes for teacher  TODO: Homework exercises  62 | | |
| **Day 3: Merge Sort**  Code Merge Sort for students and perform complexity analysis on the algorithm. Have the code ready to go with the StopWatch class so we can compare it with one of the quadratic sorts. Students will not write this algorithm.  **Homework:** None  TODO: Lecture notes for teacher  63 | | |
| **Day 4: Recursion Lab**  Pass out the Towers of Hanoi lab. Students should spend the first part of the period understanding the problem and writing the code on paper. They should then work in pairs to code it on a computer.  **Homework:** None  TODO: Lab sheet  64 | | |
| **Day 5**  Flexible. However, if there is time, students can code the four sorts and use the StopWatch.java class to time them and come up with regression equations like they did with the two searches. It is particularly interesting to compare the three quadratic sorts.  TODO: Optional activity | | |

|  |  |  |
| --- | --- | --- |
| Week 15: P/NP/NP Complete |  | Quarter 2 |
| **Weekly Overview** | |  |
| This week presents, in light of previous material, the first real limitation of a computer. This is a practical limitation, not a theoretical limitation. The idea is that some things *can* be coded, but their best-known algorithms are too slow to be useful. Students will learn about the categories of P, NP, and NP Complete, and understand that this is the number one unsolved problem in computer science. This looks ahead to heuristics (what do you do if you cannot write an algorithm because either (a) you don’t have the information, or (b) it will take too long). Finally, students will program an NP Complete problem to demonstrate that (1) the algorithm is not difficult, but (2) it takes way too long to run for large cases. | |  |
| **Main Objectives** | |  |
| * Describe the categories of P, NP, and NP Complete * Program an NP Complete Problem. * Extrapolate how long it would take to compute the solution to the NP Complete problem for large cases. | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  NP Complete Programming Lab | | |
| **Extra Teacher Resources** | | |
| **Readings**  “Intractability”, *Computer Science: An Interdisciplinary Approach*: <https://introcs.cs.princeton.edu/java/55intractability/>  “P vs. NP and the Computational Complexity Zoo”: <https://www.youtube.com/watch?v=YX40hbAHx3s> | | |
| **Daily Outline** | | |
| **Day 1: Introduction to P/NP**  This is a lecture on the categories of P and NP. We will not yet mention NP Complete. However, students need to see that the question of whether P = NP is the biggest open question in computer science. Introduce at least two NP Complete problems without calling it by that name. Emphasize that the algorithm is known, but it is too slow.  Mention RSA Encryption and how it depends on not being able to factor very large numbers efficiently. Make the points that our nations security depends on this fact. Also make the point that simply getting faster computers will not help. Assuming exponential algorithms, explore what happens as single digits are added to 100-digit long composite numbers.  **Homework:** None  65 | | |
| **Day 2: NP Complete**  This is another course “climax”. Introduce the concept of NP complete, and then give a couple more famous problems.  **Homework:** None  66 | | |
| **Day 3: NP Complete Lab**  The first part of this lab should be programming a simple NP Complete Problem.  **Homework:** None  67 | | |
| **Day 4: NP Complete Lab**  The second part of this lab is to time the algorithm on reasonable data, and use the efficiency analysis to extrapolate how long it would take for bigger cases.  **Homework:** None  68 | | |
| **Day 5: Test**  Test on recursion and complexity analysis.  TODO: Test?  69 | | |

|  |  |  |
| --- | --- | --- |
| Week 16: Objects |  | Quarter 2 |
| **Weekly Overview** | |  |
| This week contains the second abstraction: Objects. In “methods”, students abstracted “verbs”. Here they abstract “nouns” by writing their own data types. Students will generate their first data structure: the ArrayList, which is like an array, but has no bound on size, no gaps, and can have elements inserted and deleted. This is laying the groundwork, too, for inheritance and polymorphism. | |  |
| **Main Objectives** | |  |
| * Write an Object. * Understand how parameters work with Object references. * Understand the difference between static and non-static methods. * Write the ArrayList Object using an array. | | |
|  | | |
|  | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1: Objects**  Objects are different than primitives. Students need to review how arrays work and why they are different when they are passed to methods. This first lecture should also introduce the idea that you can write your own Object, and you should work together to construct a simple example in class, complete with a constructor, mutator, accessor, and some other interesting methods. A discussion about static versus non-static methods should also take place. When they see a method for the object, it is a good time to describe the difference between a static and a non-static method. Until now, they were always using “static” before all of their methods.  **Homework:** Add a method to the class we developed.  TODO: Lecture notes for teacher  TODO: Homework sheet for students  70 | | |
| **Day 2: Objects (continued)**  It is likely true that the previous lesson will spill into this day. Today should also include exercises on how Objects act when they are passed to methods.  **Homework:** Outline an ArrayList object and have students write the constructor, set(), and get().  TODO: Lecture notes for teacher (potentially includes in yesterday’s notes)  TODO: Homework sheet for students  71 | | |
| **Day 3: ArrayList Lab**  Students should begin writing ArrayList. The methods will include: get(), set(), delete(), insert(), and toString(), along with constructors. They will have to think about resizing the array. Provide them with a test file and test code that they can use to test their class.  **Homework:** None  TODO: Lab sheet  72 | | |
| **Day 4: ArrayList Lab**  Finish up the lab.  **Homework:** None  73 | | |
| **Day 5: Test**  Test on Objects  TODO: Test?  74 | | |

|  |  |  |
| --- | --- | --- |
| Week 17: Hardware |  | Quarter 2 |
| **Weekly Overview** | |  |
| NEED  Teacher Note: INSERT NOTE ABOUT THE POSSIBILY OF CONDENSING TO ONE DAY OR ELIMINATING IF THERE IS A LACK OF TIME | |  |
| **Main Objectives** | |  |
| * NEED | | |
|  | | |
|  | | |
| **Required Student Resources** | | |
| **Readings**  Need | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1**  **Homework:**  75 | | |
| **Day 2:**  **Homework:**  76 | | |
| **Day 3:**  **Homework:**  77 | | |
| **Day 4:**  **Homework:**  78 | | |
| **Day 5: Flexible**  On the last day of the week, pass out the exam sheet. Students are able to work on this at home, but they may *not* bring notes or code with them into class. They will have a short opportunity to ask clarifying questions about the nature of a Stack just before the exam begins. | | |

|  |  |  |
| --- | --- | --- |
| Week 18: Semester Exam |  | Quarter 2 |
| **Weekly Overview** | |  |
| This week is exam week. | |  |
| **Main Objectives** | |  |
| * Take the semester exam. | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  None | | |
| **Extra Teacher Resources** | | |
| **Readings**  None | | |
| **Daily Outline** | | |
| **Exam Day**  The exam will be to program the Stack. You will need to spend 15 minutes explaining the concept and giving examples of how the structure works by calling specific methods. Provide them with the code and the exam sheet. It is due at the end of the period. | | |

|  |  |  |
| --- | --- | --- |
| Week 19: A Return to Aristotle |  | Quarter 3 |
| **Weekly Overview** | |  |
| At the start of the second semester, we return back to Aristotle and his categorization of the syllogism. This is to serve two purposes. First, in placing syllogism here, we give a much needed reminder of (1) the distinction between formal and material logic, and (2) the fact that a computer cannot “know” or “decide.” Second, it sets the stage for a return to truth trees, which, together with polymorphism, will form the basis of the upcoming lab.  Students will work with the syllogisms of the first figure by reading Aristotle’s descriptions. They will diagram these figures with Venn Diagrams, and will be given the rest of the figures as a worksheet. Students will explore hypothetical syllogism as well – these form a philosophical basis for the conditionality of certain information and conclusion. | |  |
| **Main Objectives** | |  |
| * Work in detail with syllogisms of the first figure. * Explore syllogisms in all moods and figures. * Work with hypothetical syllogisms. | | |
| **Required Student Resources** | | |
| **Readings**  *Prior Analytics,* I.2–I.4, Aristotle. | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1: The First Figure**  Have the students read together *Prior Analytics,* I.2–I.4 and work to categorize the syllogisms of the first figure. Remind them at the end of the distinction between formal and material logic, and help them diagram the syllogisms using sets, providing one as a class example. At the end of the lesson, students should be given a list of all syllogisms in all moods and figures.  **Homework:** Syllogism exercises.  TODO: Teacher Lecture Notes  TODO: Homework Assignment  1 | | |
| **Day 2: The Rest of the Figures**  Use the listing from yesterday to take students through the other figures, and them practice.  **Homework:** More syllogism exercises.  TODO: Teacher Lecture Notes  TODO: Homework Assignment  2 | | |
| **Day 3: Syllogism Practice**  This day is reserved for practice using the Aristotelian figures.  **Homework:** More syllogism exercises  TODO: In-class Exercises  TODO: Homework Assignment  3 | | |
| **Day 4: Hypothetical Syllogisms**   1. Introduce hypothetical syllogisms. 2. Practice   **Homework:** Exercises on hypothetical syllogisms  TODO: Teacher Lecture Notes  TODO: Homework Assignmen  4 | | |
| **Day 5**  Flexible | | |

|  |  |  |
| --- | --- | --- |
| Week 20: A Return to Aristotle |  | Quarter 3 |
| **Weekly Overview** | |  |
| Students will continue their work on logic by working with conjunctive and disjunctive syllogisms, and then culminating in a general discussion of important logical fallacies. Significant time is left this week to allow for practice as well as an assessment. | |  |
| **Main Objectives** | |  |
| * Work with Conjunctive and Disjunctive Syllogisms. * Identify logical fallacies, including equivocation, undistributed middle term, self-referential pseudo-propositions, *ad populum*, *ad hominem*, and *post hoc ergo propter hoc.* * Understand the division of terms and their related fallacies. | | |
| **Required Student Resources** | | |
| **Readings**  Need (Isagoge) | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1: Conjunctive and Disjunctive Syllogisms**   1. Introduce conjunctive and disjunctive syllogisms. 2. Practice   **Homework:** Exercises on conjunctive and disjunctive syllogisms  TODO: Teacher Lecture Notes  TODO: Homework Assignment  5 | | |
| **Day 2: Hypothetical, Conjunctive, and Disjunctive Syllogisms Practice**  This day is reserved for more practice on hypothetical, conjunctive, and disjunctive syllogisms.  **Homework:** More exercises on hypothetical, conjunctive, and disjunctive syllogisms  TODO: In-class Exercises  TODO: Homework Assignment  6 | | |
| **Day 3: Fallacies**  The division of terms and related fallacies, Propositions, Reasoning (Equivocation, Undistributed middle term, Self-referential pseudo-propositions, *ad populum*, *ad hominem*, *post hoc ergo propter hoc*).  **Homework:** Exercises on fallacies  TODO: Teacher Lecture Notes  TODO: Homework Assignment  7 | | |
| **Day 4: Review**  Because of the tight nature of logic at the start of this semester, we are allotting an extra day for practice and review of the concepts prior to the test. If the prior day’s material on fallacies runs over, this day can also be sued to catch up on that.  **Homework:** None  TODO: In-class review exercises?  8 | | |
| **Day 5: Test**  Test on second semester logic material  **Homework:** Read the *Isagoge* reading.  TODO: Test?  9 | | |

|  |  |  |
| --- | --- | --- |
| Week 21: Inheritance and Polymorphism |  | Quarter 3 |
| **Weekly Overview** | |  |
| This week focuses on inheritance and polymorphism and sets the stage for the close of the semester. Students will first discuss Porphyry’s *Isagoge*, which provides a beautiful parallel with the hierarchy of inheritance in code. Students will write a set of classes with inheritance, and will explore how polymorphism impacts the execution of methods at run time. They will make a distinction between compile time reference type and run time construction type. | |  |
| **Main Objectives** | |  |
| * Understand Porphyry’s tree of being. * Write a series of classes with an inheritance hierarchy. * Understand how polymorphism impacts how an object acts at run time. * Understand the difference between compile-time reference type (checked trough static analysis) and runtime construction type. * Understand the use of Java generics and the relationship with polymorphism. | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1: Seminar on Categories**  Conduct a seminar on the *Isagoge* excerpt.  TODO: Seminar questions and discussions for the *Isagoge* text  10 | | |
| **Day 2: Inheritance**  Introduce inheritance with a basic example. The point at first is to emphasize code reuse. If we reuse code, the class becomes simpler. At the end of this lecture introduce the Java Object class and its methods, especially toString().  **Homework:** Simple inheritance exercise.  TODO: Lecture notes for teacher  TODO: Homework exercises  11 | | |
| **Day 3: Polymorphism**  This is a major lecture and may be difficult for students. Introduce the idea and take them through various scenarios of (1) what causes a compile error, and (2) what method actually gets executed if the code passes the compile stage. They will need to see various examples. At the end, note that toString works (in things like System.out.println) *because* of polymorphism (because al classes inherit from the Object class, which contains the method).  If there is time, introduce the concept of an Abstract Class.  **Homework:** Polymorphism Exercises  TODO: Lecture notes for teacher  TODO; Homework exercises  12 | | |
| **Day 4: Polymorphism Examples**  This day should be spent going over the exercises and providing more examples.  **Homework:** None  TODO: In class exercises  13 | | |
| **Day 5**  Flexible | | |

|  |  |  |
| --- | --- | --- |
| Week 22: Polymorphism Lab |  | Quarter 3 |
| **Weekly Overview** | |  |
| This week is a robust lab experience involving inheritance and polymorphism. | |  |
| **Main Objectives** | |  |
| * Apply inheritance and polymorphism to solve problems. | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1: Polymorphism Lab**  Pass out the code to the “Inheriting Inheritance” lab. Because this lab involves graphics, students will need to be talked through the code. The good thing is that the graphics themselves emphasize polymorphism. The lab is to create classes for Creature, Animal, and Dog, together with a hierarchy of genes that approximate simple, incomplete, and co-dominance and run simulations of a group of 100 offspring based on their parents’ genes.  **Homework:** None  TODO: Code for lab  TODO: Lab sheet  14 | | |
| **Day 2: Polymorphism Lab**  Continued work on lab.  **Homework:** None  15 | | |
| **Day 3: Polymorphism Lab**  Continued work on lab.  **Homework:** None  16 | | |
| **Day 4**  Flexible  17 | | |
| **Day 5: Test**  Test on Inheritance and Polymorphism  TODO: Test?  18 | | |

|  |  |  |
| --- | --- | --- |
| Week 23: The Parse Tree Logic Lab |  | Quarter 3 |
| **Weekly Overview** | |  |
| This week is a major programming lab that brings together formal logic (truth trees), inheritance, polymorphism, data structures, and recursion. Should we build in truth table material, or even transfer some from S1 here? | |  |
| **Main Objectives** | |  |
| * Apply inheritance, polymorphism, and recursion to model truth trees. | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1: A Binary Tree**  Define a binary tree, and use this to talk about truth trees. This will involve a return to the logic of Quarter 1. Pass out existing code that implements a truth tree (using links, I think). This will be the code on which the students’ lab is based. Spend this period talking through the code. It is not necessary to talk just yet about links. Can we build in dead branches and dead trees into this?  **Homework:** Write a method that calculates the number of nodes in the tree. Hint: think recursively.  TODO: Lecture notes for teacher  TODO: Homework sheet  19 | | |
| **Day 2: Truth Tree Lab**  Students will need to write the various types of Nodes for the tree. The actual tree class will also required additional functionality. At a minimum, this should involve evaluate(), print(), depth(), numberOfLeaves(), numbersOfNonLeaves(), and numberOfNodes(). This lab brings together logic trees, objects, inheritance, polymorphism. This lab will also provide an API where students can specify the value of variables, and their evaluate() code will be invoked to compute the results of a statement in the language. The lab provides a parser for students for statements in the language.  **Homework:** None  TODO: Code for lab  TODO: Lab sheet  20 | | |
| **Day 3: Truth Tree Lab**  **Homework:** None  21 | | |
| **Day 4: Truth Tree Lab**  **Homework:** None  22 | | |
| **Day 5: Truth Tree Lab**  **Homework:** None  23 | | |

|  |  |  |
| --- | --- | --- |
| Week 24: Stacks |  | Quarter 3 |
| **Weekly Overview** | |  |
| This week focuses on the Stack data structure, but also introduces the notion of Nodes. Students will have been exposed to the idea during the Tree lab, but there they will have just bee given the and not asked to work with them. In writing the Stack this way, this also gives students a way to see two different implementations of the same interface – they first being the array-based Stack that they saw during the semester exam. Finally, students will use the Stack to solve a programming problem. | |  |
| **Main Objectives** | |  |
| * Implement and work with linked nodes. * Implement a node-based Stack. * Use Stacks to solve problems. | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1: Define Stacks and Linked Nodes**  Review the Stack interface, and begin the discussion about nodes and links. This is difficult for students. One of the side goals to this is that the same data structure can have multiple implementations. Do a full complexity analysis on this, and compare it with the array-based implementation. Emphasize the lack of the need to resize with the link-based Stack.  **Homework:** If necessary, complete the methods for the Stack that were not completed in class.  24 | | |
| **Day 2: Using Stacks**  Outline the upcoming lab, which is using a Stack to figure out whether a circuit box it routable. The teacher will talk through the circuit problem, but then students will begin implementation.  **Homework:** None  25 | | |
| **Day 3: Stack Lab**  The lab will include coding the Stack itself, but also implementing the circuit routing problem.  **Homework:** None  26 | | |
| **Day 4: Stack Lab**  Finish the Stack Lab  **Homework:** None  27 | | |
| **Day 5**  Flexible | | |

|  |  |  |
| --- | --- | --- |
| Week 25: Hash Tables |  | Quarter 3 |
| **Weekly Overview** | |  |
| This week focuses on the definition of a Hash Table. The first part is an introduction to the idea, and the second part is the exploration of HashMap in Java by way of a short lab. These ideas will continue into next week with a larger lab. | |  |
| **Main Objectives** | |  |
| * Introduce a complex data structure. * Introduce the idea of a hash function and the mathematical properties of a good hash function. * Introduce techniques for handling collisions and techniques for accepting collisions when they do happen. | | |
| **Required Student Resources** | | |
| **Readings**  *Anything?* | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1: The Hash Table**  Go over the basics of what a has table is, including a discussion of hash functions.  **Homework:** Need  TODO: Lecture notes for teacher  TODO: Homework exercises  28 | | |
| **Day 2; The Hash Table (continued)**  Continued content from Day 1  **Homework:** Need  TODO: Lecture notes for teacher  TODO: Homework exercises  29 | | |
| **Day 3: Hash Table Lab**  Pass out the Hash Table Lab STILL NEED.  **Homework:** None  TODO: Lab idea  TODO: Potentially lab code  TOTO: Lab sheet  30 | | |
| **Day 4: Hash Table Lab**  Second day on Hash Table Lab.  **Homework:** None  31 | | |
| **Day 5: Test**  Test on Hash Tables and Stacks  TODO: Test?  32 | | |

|  |  |  |
| --- | --- | --- |
| Week 26: Heuristics |  | Quarter 3 |
| **Weekly Overview** | |  |
| As we get closer to the end of the year, we return to the theme of what a computer *cannot* do. We saw in the discussion of NP that some things are simply not efficient enough to be done. When this happens, we employ a heuristic rather than an algorithm. We also employ heuristics when we do not have all of the information. Students will learn the difference between a heuristic and an algorithm, and they will write a heuristic function for an intractable problem, emphasizing that *human judgment* provide the rationale for the decision. Students will change the heuristic and observe the differences. | |  |
| **Main Objectives** | |  |
| * Describe the difference between a heuristic and an algorithm. * Write a heuristic function for an intractable problem. * Observe how two different heuristic functions provide two different results in a problem. | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1**  STILL NEED TO WORK ON THIS  TODO: Fully flesh out this unit  33 | | |
| **Day 2**  34 | | |
| **Day 3**  35 | | |
| **Day 4**  36 | | |
| **Day 5**  Flexible | | |

|  |  |  |
| --- | --- | --- |
| Week 27: Heuristics Lab |  | Quarter 3 |
| **Weekly Overview** | |  |
| This week focuses on a major lab in which students are to write a heuristic to solve a complicated problem. | |  |
| **Main Objectives** | |  |
| * Apply heuristics to solve a problem. | | |
| **Required Student Resources** | | |
| **Readings**  *Anything?* | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1**  TODO: Fully flesh out this lab  37 | | |
| **Day 2**  38 | | |
| **Day 3**  39 | | |
| **Day 4**  40 | | |
| **Day 5: Test**  Test on heuristics  41 | | |

|  |  |  |
| --- | --- | --- |
| Week 28: Turing Machines, Computational Power, and von Neumann Architecture |  | Quarter 4 |
| **Weekly Overview** | |  |
| This week wraps up the progression from formal logic the working computer. Students have seen various pieces of this throughout the year, but it all comes together in the Turing Machine. Students will read from Turing directly. They will also discuss the Universal Turing Machine and writings from Babbage and Lovelace, as well as von Neumann architecture. It is setting stage for next week’s culmination on the limits of computation. | |  |
| **Main Objectives** | |  |
| * Define a Turing Machine * Describe what a particular Turing Machine accomplishes. * Design a Turing Machine to accomplish a particular task. * Define the Universal Turing Machine. * Describe the von Neumann architecture. | | |
| **Required Student Resources** | | |
| **Readings**  “On Computable Numbers, With an Application to the Entscheidungsproblem,” A. Turing (section needed for TM only)  “Sketch of *The Analytical Engine* Invented by Charles Babbage”, L.F. Menabrea with notes by A. Lovelace (section needed) | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  Need? | | |
| **Daily Outline** | | |
| **Day 1: Turing Machines**  This lecture is on the definition of a Turing Machine, how it comes out of regular expressions and DFA, and the fact that it adds memory to those objects. The emphasis is that the Turing Machine is the theoretical model on which the modern computer is built. Students should see at least a couple examples of Turing Machines and be asked to trace through “what they do”.  **Homework:** Give the Turing Machine that converts a unary number on the tape and produces the binary equivalent. Ask students to determine what it does.  TODO: Lecture notes for teacher  TODO: Homework exercises  42 | | |
| **Day 2: Turing Machines (continued)**  More examples of Turing Machines.  **Homework:** Read “On Computable Numbers, With an Application to the Entscheidungsproblem,” A. Turing (section needed for UTM only)  TODO: Decide on excerpt for the Turing reading  TODO: Lecture notes for teacher (might be a continuation of the previous day)  43 | | |
| **Day 3: The Universal Turing Machine and von Neumann Architecture**  This lecture is on the universal Turing Machine and the Church-Turing Thesis. This is the crowning achievement of this quarter and deserves some attention. Use the Turing reading from the night before. The lecture should also touch on the basics of the von Neumann architecture as well as the work of Lovelace (on Babbage). At this point, we really do have the modern computer.  **Homework:** Read “Sketch of *The Analytical Engine* Invented by Charles Babbage”, L.F. Menabrea with notes by A. Lovelace (section needed)  TODO: Lecture notes for teacher  44 | | |
| **Day 4**  Flexible. However, this week may require an extra day. The first three days can be stretched at the teacher’s discretion into this fourth day. | | |
| **Day 5**  Flexible | | |

|  |  |  |
| --- | --- | --- |
| Week 29: The Limits of Computation |  | Quarter 4 |
| **Weekly Overview** | |  |
| This week culminates in the Halting Problem, which definitively sets limits on the power of computation, and makes at least the start of an argument against “Artificial Intelligence.” | |  |
| **Main Objectives** | |  |
| * Understand the Halting Problem and its ramifications for computing. * Understand the variations on the Halting Problem, especially the inability to evaluate a logical statement as universally true. | | |
| **Required Student Resources** | | |
| **Readings**  *I feel like we should have something.* | | |
| **Assignments**  These are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1: An Attempt at the Halting Problem**  Students should be given that task of scanning a for loop and trying to decide if it is infinite. Allow them to wrestle the entire period with this, and allow them to ask questions about what they can assume. They may work in “pseudo code.” Do not reveal for them that this is not possible.  **Homework:** None  TODO: Activity sheet for students  TODO: Maybe some notes for the teacher  45 | | |
| **Day 2: The Halting Problem Solved (or not)**  Present the proof of the Halting Problem. Guide students through this by presenting only some of the steps. Done correctly, this should take the entire period.  **Homework:** None  TODO: Lecture notes for teacher  46 | | |
| **Day 3: Ramifications of the Halting Problem**  Discuss the variations of the Halting Problem, most importantly that Turing proved that there is no algorithm that takes as input a logical statement (even a formal one) and answers whether it is universally true. Connect this to the Hilbert millennium problem, as well as the work of Gödel. Serious conversation should be had about the ramifications of all of this, particularly when it comes to understanding “mind” and “judgment.’ Students should be able to connect this back to Aristotle.  **Homework:** None  TODO: Lecture notes for teacher  47 | | |
| **Day 4**  STILL NEED TO WORK THIS OUT. IF WE HAD A READING, IT WOULD HELP  TODO: What do we do this day?  48 | | |
| **Day 5: Test**  Test on the limits of computation  TODO: Test?  49 | | |

|  |  |  |
| --- | --- | --- |
| Weeks 30–34: Final Project |  | Quarter 4 |
| **Weekly Overview** | |  |
| More to come. | |  |
| **Main Objectives** | |  |
| * Apply the knowledge of the course to produce a major programming project. | | |
|  | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  None | | |
| **Extra Teacher Resources** | | |
| **Readings**  None | | |
| **Daily Outline** | | |
| **Day 1: Introduction to the Project**  Introduce the project and the project teams. STILL NEED MORE  TODO: (Major) Lab sheet, code, teacher directions, project structure, etc.  50–72 | | |

|  |  |  |
| --- | --- | --- |
| Week 35: Man and Technology |  | Quarter 4 |
| **Weekly Overview** | |  |
| More to come. | |  |
| **Main Objectives** | |  |
| Describe the difference between formal and material logic. | | |
|  | | |
|  | | |
| **Required Student Resources** | | |
| **Readings**  *???* | | |
| **Assignments**  The are homework assignments: | | |
| **Extra Teacher Resources** | | |
| **Readings**  The following readings …. | | |
| **Daily Outline** | | |
| **Day 1**  This needs to be built out. Postman? Marshall McLuhan? AI? Reading, Seminar, basically bringing it all together in a profound way.  TODO: Find a reading  TODO: Allot days (do we need the full week)?  TODO: Seminar questions and discusion points  TODO: Fill in this week  73 | | |
| **Day 2**  74 | | |
| **Day 3**  75 | | |
| **Day 4**  76 | | |
| **Day 5**  Flexible | | |

|  |  |  |
| --- | --- | --- |
| Week 36: Final Exam |  | Quarter 4 |
| **Weekly Overview** | |  |
| This week has final exams. | |  |
| **Main Objectives** | |  |
| * Take the Final Exam. | | |
| **Required Student Resources** | | |
| **Readings**  None | | |
| **Assignments**  None | | |
| **Extra Teacher Resources** | | |
| **Readings**  None | | |
| **Daily Outline** | | |
| **Exam Day**  Take the final exam. What is this?  TODO: Do we write an exam or leave it to the teacher? | | |