CSE120 Wi19 Final Review

Note: You can bring two double-sided, letter-sized (8.5” × 11”) sheets of handwritten notes to the exam.

Lecture Main Ideas

Computer Science Principles has covered two kinds of information:

1) Ideas and details concerning programming, such as arrays and Strings
2) Ideas about computational topics, such as algorithmic complexity and artificial intelligence

The final covers both of these kinds of information, including all the lecture material so far:

13. **Arrays:** *Arrays* are structures that store many values of the same datatype. You can think of them as collections of variables with special names: the array name along with the *index* of the *element* you want. Arrays are a way to associate a number (the index) with the stored data type (the element) and are often used in conjunction with loops.

14. **Images and Strings:** *Images* are 2-dimensional sets of color data. In Processing, you can load images into `PImage` variables in your program and you can read and write to your canvas’ color data using the `pixels[]` array. *Strings* are collections of characters (*e.g.* words, symbols, sentences). They can be combined using the concatenation operator (+). You can retrieve an individual character (`str.charAt(i)`) or the String length (`str.length()`). String literals are defined between double-quotes and the empty string has zero characters in it (`""`).

15. **Artificial Intelligence:** Difficult to define, but could be the science of making machines that (1) think like people, (2) act like people, (3) think rationally, or (4) act rationally. Most modern AIs are specialized to a specific task (artificial *narrow* intelligence), rely on probability and statistics to make decisions, and are applied to a wide variety of different applications. AI raises a ton of ethical issues as we place more and more decision-making in the hands of machines.

16. **Buttons & Boards:** Define clickable regions on canvas using conditionals and arithmetic. Store game “state” using variables and arrays. Use some numerical encoding to represent board game states. *Functional abstraction* to manage complexity and *testing statements* to check your code as you work.

17. **Computer Science and Ethics:** *Computer ethics* includes consideration of both personal and social policies for the ethical use of computer technology, which can be informed by how we choose to *conceptualize* that piece of technology. As new technologies are built, we must consider whose and what *values* we are advancing and whose values we are overlooking or even harming. Users and technologists have agency over which technologies are built.

18. **Human-Computer Interaction:** Bad design can hurt and good design is hard. *Participatory Design* is an iterative process that involves users because YOU ARE NOT THE USER. *Contextual*

Inquiry is a process of learning your users' needs and habits by observing them using a product in a naturalistic setting. *Low Fidelity Prototyping* is a tool to help with contextual inquiry where you create a quickly-sketched storyboard of your program to gather feedback on your design early on in your design cycle.

19. **Timing, Algorithmic Complexity:** Needs metrics to compare different algorithms. One common metric is computation time. *Orders of growth* allow us to characterize algorithms by how their runtimes vary by input size. We can sometimes increase our performance through a technique called *memoization*, where we remember previous calculations so we don't need to recalculate them every time.

20. **Machine Learning:** A *supervised learning* algorithm builds a *classifier* using *training data*, which are a set of known, labeled examples. It then uses that classifier to label new, unknown examples, sometimes updating its classifier based on the outcomes. The training data can introduce *bias* and *variance* into the algorithm. Bias can also result from the design of the task and/or the program.

21. **Computers:** A computer has five main components (*control*, *datapath*, *memory*, *input*, and *output*) and is a combination of hardware and software, managed by an *operating system*. The CPU interprets instructions using a fetch and execute cycle, which is a mechanical and deterministic method of performing instructions in order. You will not be tested on this lecture.

22. **Digital "Reality":** The abundance of information (and bullshit) allows us to choose what we believe in because we often end up in biased *filter bubbles* and seek out opinions that corroborate our own. This can either polarize people or make them more tolerant. This has also led to the proliferation of *conspiracy theories*, which sites often try to deal with via *content moderation*. Content can be spread more quickly online by evolving over time to play on our emotions more or via *advertising* and as *sponsored content*.

23. **Security:** Use a *security mindset* to think like an attacker in order to improve the security and privacy of technologies: analyze the different assets and how they might be attacked, then build new defenses for any vulnerabilities that are found. You will not be tested on this lecture.

24. **Limits of Computing:** We can classify computing problems into "easy" (in P, with polynomial-time solutions), "hard" (not in P, slower than polynomial time to solve and verify), and maybe hard (in NP, slow to solve for now, but easy to verify). A *decision problem* has a yes/no answer. A *decidable* problem has a solution. Despite the wonders of computing, some problems are undecidable (*i.e.* can't be solved by computers).

25. **Computing for Social Good:** *Accessibility* is the ability for someone to use a service or tool. Accessibility development for those with disabilities usually benefits everyone. Computing and technology can be a huge boon to people and areas that don't typically have access to them. However, when designing tech for people/users, you have to take into account their social and cultural contexts as well as environment. *Crowdsourcing* utilizes a variety of motivated agents to solve a problem in a distributed manner. You will not be tested on this lecture.

Practice Questions

Completing these questions will help you prepare for the final, but please note that this list does not cover all material that may be on the final, but instead is a guide to help you review. Completing these on a separate piece of paper is suggested, as extra space is not provided here. It is highly recommended to write out code on *paper* first to give you practice for the exam. Then you can type your answers into Processing to test and verify. Feel free to work together!

Questions will be based on the lectures, section worksheets, and assignments:

1. True or false? Looping is necessary for complex programs. Briefly explain.

2. Briefly explain the benefit of using *parameters*. What is the *variable scope* of parameters?

3. Briefly explain the benefit of using *arrays*. Name a few examples of array usage in this class.

4. Calculating exam statistics takes time! If was assume that all of the scores for the final are stored in an array, *describe in words* (not code) algorithms to calculate the MEAN (average) and MEDIAN (middle score). Your answers should reference the array.

5. We want to use an array of colors of length 6. Write Processing *code* below that declares such an array (use any name of your choosing) and initializes them to different shades of blue evenly spaced from black to totally blue.

6. Write a Processing *function* called `grid()` that draws a grid of rectangles of width 50 and height 20. The user should be able to specify the coordinate of the upper-left corner of the grid as well as the number of rows and columns of rectangles.

7. Write a Processing *function* called `hasX()` that returns `true` if the `String` that is passed to it contains the character `'x'` in it and `false` otherwise.

8. Write a Processing **`draw`**`()` function that draws green squares of size 50 along the diagonal of your drawing canvas, starting from the upper-left and going towards the lower-right. The squares should touch neighboring squares at their corners.

9. Write a Processing **`draw`**`()` function that draws a thick, horizontal, black line of length 100 that is centered in the middle of your drawing canvas. This line should follow your mouse vertically (but not horizontally).

10. What gets printed to the console when the following program is run?

```
String soda = "Pepper";
String pop = "Coke";
println("Is it " + soda.equals(pop) + " that Dr " + soda + " has " +
2 + soda.length() + " flavors?");
```

11. Write a `keyPressed()` function that detects the number keys (0-9) and prints the associated symbols found above them on the keyboard (*e.g.* 1 → `!`, 2 → `@`, 3 → `#`) to the console.

12. Complete the following function that prints a triangle of asterisk characters of a specified height to the console. Some examples:

`draw_triangle(3)` should print out:

```
*
***
*****
```

`draw_triangle(4)` should print out:

```
*
***
*****
*******
```

```
void draw_triangle(int height) {

  for ( int i = 0; i < __________; i = i + 1) {

    for ( int j = 0; j < __________; __________ ) {

      print( __________ );
    }
    println();  // starts a new line
  }
}
```

13. Write a Processing *program* that constantly draws colored lines from the position of the last mouse click to the current mouse position. Initially, the "last mouse click position" should be `(0,0)`. Each time the mouse is pressed, the drawing canvas should be cleared and the line color should change (use a rotation of red → green → blue). New lines will be drawn from this new position to the current mouse position.

14. Fill in the blanks to complete the function `twoSum()` below, which takes in a non-empty array and a value and returns whether or not there exist two *different* elements (*i.e.* different locations, though values could be the same) in the array that sum to the specified value. Examples:

`arr: {1,5,7,2,3}, val: 10` → `twoSum(arr, val)` returns `true` because 7 + 3 = 10

`arr: {2,4,9,2,0}, val: 7` → `twoSum(arr, val)` returns `false`

```
__________ twoSum( int[] arr, int val ) {

   for ( int i = 0; i < __________; i = i + 1 ) {

      for ( int j = 0; j < arr.length; __________ ) {

         if ( (i ____ j) && (__________ == val) ) {

            return __________;
         }
      }
   }
   return __________;
}
```

15. Fill in the blanks below to complete the function `arrayConcat()`. This function takes in two non-empty integer arrays and returns a single integer array that is the two arrays concatenated (put together). `arr1` should be *before* `arr2` in the returned array. This could be used to help Elli grow after eating an apple! Examples:

`arr1: {1,1}, arr2: {2,2,2}` → return **`{1,1,2,2,2}`**

`arr1: {80},  arr2: {100,120,140,160}` → return **`{80,100,120,140,160}`**

```
_________ arrayConcat(int[] arr1, int[] arr2) {

    int newLength = _____________ + _____________;
    int[] concatArr = new int[newLength];
    // copy over arr1 into front of concatArr

    for ( int i = 0; i < _____________; i = i + 1 ) {

        concatArr[ _________ ] = arr1[ _________ ];
    }

    for ( int i = 0; i < arr2.length; i = i + 1 ) {
        // how many spots have already been filled?

        concatArr[ ________________ ] = arr2[i];
    }

    ___________________________;

}
```

16. Fill in the blanks below to complete the function `charsToString()`. This function should take in a character array and return a `String` that contains the characters in the array in order. Examples:

`arr: {'h','e','l','l','o'} → charsToString(arr)` returns **`"hello"`**

`arr: {'I',' ','<','3',' ','U'} → charsToString(arr)` returns **`"I <3 U"`**

```
________ charsToString( ________ arr) {

    ________ output = "";

    for ( _________; ____________; i = i + 1 ) {

        output = ____________________;
    }

    return __________;
}
```

17. Most modern video game consoles not only play games and movies, but have cameras, microphones, and online stores. Thinking about the security of such a system, name two different assets that an attacker might go after and what the attacker might do with that stolen information.

18. A computer that can pass the Turing Test accomplishes which goal of AI? (Circle one)

Think like people   Act like people   Think rationally   Act rationally

19. Briefly explain what the Filter Bubble effect is and why it is problematic.

20. Which is usually larger in size: a text file or a video file? Briefly explain why.

21. What do the acronyms **HCI** and **UX** stand for? Why are these important?

22. Name one machine learning/artificial intelligence/game theory **algorithm** mentioned in the lecture AND a **computation problem** that that algorithm can be applied to.