CSE120 Wi19 Midterm Review

Note: You can bring one double-sided, letter-sized (8.5" × 11") sheet 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 variables and for-loops
2) Ideas about computational topics, such as abstraction and binary

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

1. **Abstraction:** A technique for dealing with complexity by *generalization* and/or *removal of details*. In computer science, we break complex problems into smaller, more manageable, and reusable tasks.

2. **Lightbot and Functions:** Lightbot 2.0 seemed like a game, but it was also *programming*: the act of preparing *instructions* that an *agent* follows to achieve a specific *goal*. *Functional abstraction* is the act of identifying a sequence of operations that achieve a significant goal and giving them an identifiable *name*.

3. **Binary:** You can represent anything countable using just 0's and 1's (bits and bytes)! How the binary number system works, including conversion into other bases (decimal and hexadecimal).

4. **Processing and Drawing:** Processing is text-based, so we write down instructions and the computer follows them when we execute a program. Learned some basic syntax, including semicolons, *colors* (RGB), and *shapes*.

5. **Variables & Datatypes:** A *variable* is a piece of your program that holds a value. You *declare* a variable by specifying its name and *datatype* (`int`, `float`, `color`, `boolean`, etc.), but then need to *initialize* the variable to some initial value before using it. You can modify the value in a variable throughout your program. Also learned about the `min()` and `max()` functions.

6. **Algorithms:** A *computational problem* specifies a desired input/output relationship. An *algorithm* describes a procedure to solve a computational problem. An *implementation* is an algorithm expressed in a way that a computer/agent can execute it. There are many different possible algorithms for every computational problem.

7. **Functions in Processing:** *Functions* are defined once, but called as many times as desired. A function declaration includes a *return type* (datatype), name, and list of *parameters* (internal variables). Parameters allow different calls to a function to vary by passing the *argument* values of the function call into the parameter variables, which are used in the function *body*. *Active mode* in Processing uses the **`setup`**`()` and **`draw`**`()` functions.

8. **The Internet:** A *network* is a group of computing devices connected together. *The Internet* is the largest network of networks on the planet and was built to be *decentralized* and using *open standards*. Specific devices can be reached using the *domain name system* and *IP addresses*. Data is secured when passed on the Internet using *encryption*.

9. **Expressions & Conditionals:** An *expression* is a combination of values, variables, operators, and functions that produce another value. Operators are built-in "functions" that use special symbols. Control flow is the order in which instructions are executed. This is normally sequential from top-to-bottom, but is changed by function calls, *conditional statements*, and loops. If-statements (`if`, `else if`, `else`) allow for decision-making. Also learned about the modulus (%) operator.

10. **Basic Input & Output:** *System variables* are special variables that hold values related to the state of the program, including user input (e.g. `mouseX`, `mouseY`, `mousePressed`, `key`, `keyPressed`). Text output can be printed to the console or the drawing canvas. Each key press is represented by the ASCII encoding of the associated *character*.

11. **Digital Distribution:** The ability to copy digital information instantaneously and without loss has a huge legal, cultural, and economic impact on content creators. *Copyright* protects the rights of the creator to the use and distribution of his/her *intellectual property*. If the copyright holder chooses to, he/she can distribute work fully under an *open-source license* (typically applies to software) or with some restrictions using a *Creative Commons license*.

12. **Loops & Nested Loops:** Looping (`while`, `for`) is another form of control flow that allows for sequences of near-repetitions based on given conditions and update statements. Loops can be *nested* (i.e. put inside of other loops) in order to increase the number of dimensions you are varying.

13. **Arrays:** *Arrays* will not be tested until the final.

14. **Privacy:** *Privacy* will not be tested until the final.

Practice Questions

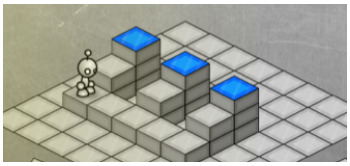
Completing these questions will help you prepare for the midterm, please note that this list does not cover all material that may be on the midterm, but instead is a guide to help you review. Completing these on a separate piece of paper is suggested, as extra space is not provided here. It is highly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

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

1. Lightbot questions
   a. How does the Lightbot keep track of where it is in the program?

   b. Lightbot (and computers) can do only a small amount of instructions; how do programmers avoid the tedium of using these instructions all the time?

   c. If `F1` is a Lightbot function that contains: `Left, Jump, Jump, Left`. Write an `F2` so the solution for the problem at right is `3:(F2)`.



2. Functional abstraction is recognizing a concept/subtask in a sequence of operations and giving it a name; why do we bother to abstract?

3. Give two ways in which the functions of Processing are "more powerful" than the functions of Lightbot – what can a Processing function do that a Lightbot function can't?

4. What is the difference between an *active* and a *static* program in Processing? Provide some examples in your answer.

5. Is the following line of code correct, incorrect, or unknown? Explain your choice.

```
line(150, 150, mousex, mousey);
```

6. `fill()` can take up to 4 arguments. In order, what do these arguments set/determine?

7. Describe the following background colors
   a. `background(255, 0, 0);`
   b. `background(64);`
   c. `background(0, 0, 64);`

8. Define the term *variable*.

9. State the data type of each of the values below:
   a. `3.14`
   b. `color(0)`
   c. `-3`
   d. `'x'`
   e. `true`

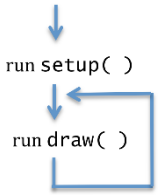
10. State the data type of each of the values returned by the system variables below:
    a. `mouseX`
    b. `width`
    c. `key`
    d. `keyPressed`
    e. `frameCount`

11. In your own words, describe what the statement below does.

```
int ra = 200;
```

12. What are the possible values that a `boolean` variable can hold?

13. Explain the diagram below in your own words.



14. Briefly explain what the following do or what they are used for:
    a. `%`
    b. `min()`
    c. `||`
    d. `;`
    e. `{` and `}`

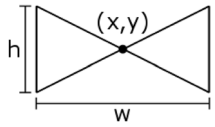
15. How many parameters does the function header below have?

```
void ninja(int x_pos, int y_pos, color goggles)
```

16. Write a piece of code that will display a circle 20 pixels wide that flashes random colors.

17. If I wanted to write a program that draws 4 identical faces on the screen, explain in words (not code) the best way to go about this. What kind of programming constructs should you use?

18. Complete the program below that draws the bowtie shown at right:




```
float x = 20, y = 20, w = 40, h = 20;

triangle( _______, _______, _______, _______, _______, _______ );

triangle( _______, _______, _______, _______, _______, _______ );
```

19. In the code below, which variables are *parameters*? Which variables are used in *arguments*?

```
color gray (int shade, int number) {
  record = max(number, record);
  return color(shade+2, shade+2, shade+2);
}
```

20. Do the two statements below mean the same thing? Explain your answer.

```
seconds = seconds + 1;

seconds + 1 = seconds;
```

21. Given the for-loop below, draw the result. Label at least one point per shape.

```
for (int i = 0; i < 4; i = i + 1) {
  rect(50, 50+20*i, 10, 10);
}
```

22. If the 4 in the code above (Question 21) is changed to a 5, what will change in the drawing?

23. Write a snippet of code equivalent to Question 21 that uses a while-loop instead of a for-loop.

24. If `void dog(float xpos, float ypos)` is a function that draws a dog at `(xpos, ypos)`, write a for-loop to draw ten dogs in a row, with a dog every 100 pixels in the *horizontal* direction.

25. We're writing a function that computes the *factorial* of an integer. The factorial of a number n is written as $n!$ and is defined as $n! = n \times (n-1) \times \ldots \times 1$. For example, $4! = 4 \times 3 \times 2 \times 1 = 24$. Also note that $0! = 1$.

    Some code is already provided for you. Your goal is to finish filling in the function:

```
int factorial(int n) {
  int product = ______;
  for (int i = ______; i > ______; i = i - 1) {
    product = product * ______;
  }
  return __________;
}
```

26. Name one benefit of digitizing information.

27. What is an Internet packet?

28. What is a "byte"?

29. The binary number `10 0101` represents what decimal number?

30. What are all the possible three-bit binary sequences?

31. Name or describe a famous algorithm that you use on a regular basis.

32. Give an example of a digital rights management (DRM) technology. Briefly explain how it is intended to uphold copyright.