1. Logic gates is a level of abstraction between microarchitecture and transistors. How exactly does logic gates connect microarchitecture to transistors?

The computer's microarchitecture is a high-level organization of circuits organized according to their various jobs. These units are expressed in terms of basic components called logic gates. Logic gates are made up of transistors, which are actual physical devices one can build.

2. Recall the layers of abstraction involved in computing, from the very highest system level all the way down to the transistor level. What layer of abstraction is described in each of the following statements?

2.1 A bunch of loads into memory, stores from memory, arithmetic computational instructions, and a branching instruction at the ISA level, is all contained in one single Python “if” statement.
2.2 This contains both the architecture of a computer as well as the instruction set for each different component.
2.3 An ALU consists of many AND and OR gates.
2.4 A very common type of 2-input NAND gate consists of 4 transistors.

1. programming language 2. ISA 3. microarchitecture 4. logic gates

3. What role does the number sign, pound sign, or hashtag character (#) play in Python? Give an example of using it.

# is the comment sign.
# this is a comment
# this will be ignored by the compiler
4. Step through this code by filling out the table below.

(i) \( x = 13 \)
(ii) while(\( x > 5 \)):
(iii) \( \text{print("hello")} \)
(iv) \( x = x - 3 \)

<table>
<thead>
<tr>
<th>Current line</th>
<th>Calculation performed by this line</th>
<th>Variables</th>
<th>Next line:</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>initializing ( x ) to 13</td>
<td>13</td>
<td>ii</td>
</tr>
<tr>
<td>ii</td>
<td>checking if ( x &gt; 5 )</td>
<td>13</td>
<td>iii</td>
</tr>
<tr>
<td>iii</td>
<td>printing hello</td>
<td>13</td>
<td>iv</td>
</tr>
<tr>
<td>iv</td>
<td>subtracting 3 from 13 and storing it back in ( x )</td>
<td>10</td>
<td>ii</td>
</tr>
<tr>
<td>ii</td>
<td>checking if ( x &gt; 5 )</td>
<td>10</td>
<td>iii</td>
</tr>
<tr>
<td>iii</td>
<td>printing hello</td>
<td>10</td>
<td>iv</td>
</tr>
<tr>
<td>iv</td>
<td>subtracting 3 from 10 and storing it back in ( x )</td>
<td>7</td>
<td>ii</td>
</tr>
<tr>
<td>ii</td>
<td>checking if ( x &gt; 5 )</td>
<td>7</td>
<td>iii</td>
</tr>
<tr>
<td>iii</td>
<td>printing hello</td>
<td>7</td>
<td>iv</td>
</tr>
<tr>
<td>iv</td>
<td>subtracting 3 from 7 and storing it back in ( x )</td>
<td>4</td>
<td>ii</td>
</tr>
<tr>
<td>ii</td>
<td>checking if ( x &gt; 5 )</td>
<td>4</td>
<td>END</td>
</tr>
</tbody>
</table>

5. Give a counterexample to this statement: *Using only letters, numbers, and underscores will always form a valid variable name.*

Anything that starts with a number is an invalid variable name.
6. You are given three variables \( a \), \( b \), and \( c \). Write a program to sort them ascending, such that \( a \) will be less than or equal to \( b \), which will be less than or equal to \( c \). Feel free to use Homework 2 Question 10 (write a program to swap values of two variables) to assist you. You may create additional variables if needed.

```python
a = 2
b = 1
c = 3

if (a >= b):
    # swap
    temp = a
    a = b
    b = temp

if (b >= c):
    # swap
    temp = b
    b = c
    c = temp

if (a >= b):
    # swap
    temp = a
    a = b
    b = temp

print a, b, c
```
7. Let say this is our state machine.

Also, let’s say that our machine is currently in the “GET NEXT INSTRUCTION” state. What state will our state machine be in after add, add, add, jump, jump, add, add?

<table>
<thead>
<tr>
<th>Action</th>
<th>start</th>
<th>add</th>
<th>add</th>
<th>add</th>
<th>jump</th>
<th>jump</th>
<th>add</th>
<th>add</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>GET NEXT INSTRUCTION</td>
<td>ADD THE TWO GIVEN NUMBERS</td>
<td>STORE THE RESULT</td>
<td>GET NEXT INSTRUCTION</td>
<td>JUMP AHEAD THE SPECIFIED NUMBER</td>
<td>GET NEXT INSTRUCTION</td>
<td>ADD THE TWO GIVEN NUMBERS</td>
<td>STORE THE RESULT</td>
</tr>
</tbody>
</table>

8. Let’s say a piece of code does not work as expected. How do you determine whether it is a syntax error or a logic error?

If it doesn’t run or compile, it’s a syntax error. If it runs but the result is not what was expected, it’s a logic error.
9. (a) What is abstraction? Why is it important?

Abstraction is the consideration of a system as a series of subsystems that interact with each other. We do not need to know the details of the implementation, only how the interface of how to use them. This allows us to consider each subsystem separately and makes it easier to debug them.

(b) Give a brief description of the following layers of abstraction:

i. programming language

A programming language is a more human-friendly way of specifying to the computer what computations to perform.

ii. microarchitecture

This is the high-level organization of computer circuit according to the job they perform.

iii. ISA

The Instruction Set Architecture is the low-level set of instructions that the computer directly understands.

10. What is an algorithm? What are trade-offs when it comes to writing an algorithm?

An algorithm is a series of steps for solving a problem. They must be unambiguous, and it must terminate. Trade-offs occur when we must sacrifice one aspect of an algorithm’s needed resource for another: for example, speed for memory utilization.

11. What is a computer? What is the difference between a stored-program device and a fixed-program device?

A computer is a machine capable of four things: I/O, storage, arithmetic, and branching. A fixed-program device is capable of executing only one algorithm. A stored-program device is capable of running different programs or instructions.
12. Write a program to compute the factorial of 4. Your final printed answer should be 24. You may create as many variables as needed.

(many variations)

input = 4
answer = 1
counter = 1
while(counter <= input):
    answer = answer * counter
    counter = counter + 1
print answer
print(answer)

13. Rewrite the following piece of code using an array

temp1 = 72.7
temp2 = 74.6
temp3 = 68.9
temp4 = 70.2
temp5 = 76.1
print("local temperatures:")
print("Station 1: " + temp1)
print("Station 2: " + temp2)
print("Station 3: " + temp3)
print("Station 4: " + temp4)
print("Station 5: " + temp5)

//using the super basic array stuff taught in the chapter
temps = [72.7, 74.6, 68.9, 70.2, 76.1]
counter = 5
i = 0
while(i<counter):
    print "Station ", i +1, ": ", temps[i]
    i = i+1
14. Write a function that accepts two numbers and return the product of the two numbers.

```python
def multiply(a, b):
    return a*b
```

15. Create a 5 x 5 multiplication table. Use the function you created in question 14 to do the multiplication. Store the products in a two dimensional array (an array of an array), and print the two dimensional array at the end.

```python
def mult(a, b):
    return a*b

nums = [[0, 0, 0, 0, 0],[0, 0, 0, 0, 0],[0, 0, 0, 0, 0],[0, 0, 0, 0, 0],[0, 0, 0, 0, 0]]

i = 1
while (i <= 5):
    j = 1
    while (j <= 5):
        nums[i-1][j-1] = mult(i,j)
        j = j+1
    i = i+1
print nums
```

17. Print the outputs of the following piece of code.

```python
def foo(x):
    print(x)
    x = 5
    return x

y = 3
print(foo(y))
print(y)
```

```
3
5
3
```
Topics List (Chapter 1)

- What is a computer?
  - Types of computers
    - Stored program
    - Fixed program
  - Functions of a computer
    - Arithmetic
    - I/O
    - Branching
    - Storage

- Abstraction
  - Benefits of abstraction
  - Layers of abstraction in a computer
    - ISA
    - Programming language
    - State machines
    - Microarchitecture
    - Logic gates
    - Transistors

- Binary representation of numbers
- What is an algorithm?
  - Properties of algorithms
- Types of errors
  - Syntax
  - Logic
  - Runtime
- State machines
- Theory
  - Computational complexity
  - Turing machines/Turing complete
Topics List (Chapter 2)
- Programming
- Efficiency vs scalability
- Python basics
  - Syntax
    - Valid variable names
  - Conditionals
    - IF/ELSE statements
    - WHILE statements
  - Variable assignments
  - Mathematical expressions
  - Strings
  - Print function
  - Comments
  - Factorial example
  - Off-by-one errors
Topics List (Chapter 3)

- Readability vs. maintainability
- Arrays
  - Creating an array
  - Accessing elements in arrays
  - Strings as arrays
- Logical operators
  - >, <
  - ==
  - !=
  - and
  - or
- Functions
  - Defining a function
  - Returning values from functions
- Scope
1. 1 of the question this course strives to answer is “What is a computer and how to build a computer?” What is the other question? (0.5 pts)
2. What is a Turing machine? (1.5 pts)
3. Variable names can only contain letters, numbers, and __________? (1 pt)
4. Abstraction layers of a computer: system, programming language, __________, computer, ________, logic gates, _________? (2 pts)
Pop Quiz Review

1. 1 of the question this course strives to answer is “What is a computer and how to build a computer?” What is the other question? How to program a computer?

2. What is a Turing machine? An idealized computer consisting of a tape, a head (to read and write to tape), and a state machine (to decide what and where to write)
Pop Quiz Review

3. Variable names can only contain letters, numbers, and __________?
   Underscores

4. Abstraction layers of a computer: system, programming language, __________, computer, __________, logic gates, __________?
   ISA
   microarchitecture
   transistors
Exam 1 Review Worksheet

17. Print the outputs of the following piece of code.

```python
def foo(x):
    print(x)
    x = 5
    return x

y = 3
print(foo(y))
print(y)```
Exam 1 Review Worksheet

4. Step through this code by filling out the table below.

(i) \( x = 13 \)
(ii) while \( (x > 5) \):
(iii) \( \text{print}("hello") \)
(iv) \( x = x - 3 \)

<table>
<thead>
<tr>
<th>Current line</th>
<th>Calculation performed by this line</th>
<th>Variables</th>
<th>Next line:</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>initializing ( x ) to 13</td>
<td>13</td>
<td>ii</td>
</tr>
<tr>
<td>ii</td>
<td>check ( x &gt; 5 )</td>
<td>13</td>
<td>iii</td>
</tr>
<tr>
<td>iii</td>
<td>print hello</td>
<td>13</td>
<td>iv</td>
</tr>
<tr>
<td>iv</td>
<td>set ( x = 10 )</td>
<td>10</td>
<td>ii</td>
</tr>
<tr>
<td>ii</td>
<td>( x &gt; 5 ), true</td>
<td>10</td>
<td>iii</td>
</tr>
<tr>
<td>iii</td>
<td>print hello</td>
<td>10</td>
<td>iv</td>
</tr>
<tr>
<td>iv</td>
<td>( x = 7 )</td>
<td>7</td>
<td>ii</td>
</tr>
<tr>
<td>ii</td>
<td>check ( x &gt; 5 ), true</td>
<td>7</td>
<td>iii</td>
</tr>
<tr>
<td>iii</td>
<td>hello</td>
<td>7</td>
<td>iv</td>
</tr>
<tr>
<td>iv</td>
<td>( x = 4 )</td>
<td>4</td>
<td>ii</td>
</tr>
<tr>
<td>ii</td>
<td>check if ((x &gt; 5))</td>
<td>4</td>
<td>v or end</td>
</tr>
</tbody>
</table>
Exam 1 Review Worksheet

7. Let say this is our state machine.

<table>
<thead>
<tr>
<th>Action</th>
<th>start</th>
<th>add</th>
<th>add</th>
<th>add</th>
<th>jump</th>
<th>jump</th>
<th>add</th>
<th>add</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>GET NEXT INSTRUCTION</td>
<td>ADD THE TWO GIVEN NUMBERS</td>
<td>store the result</td>
<td>get the next instruction</td>
<td>jump ahead</td>
<td>get next instruction</td>
<td>add</td>
<td>store the result</td>
</tr>
</tbody>
</table>
Exam 1 Review Worksheet

9. (a) What is abstraction? Why is it important?

(b) Give a brief description of the following layers of abstraction:
   i. programming language
   ii. microarchitecture
   iii. ISA
Exam 1 Review Worksheet

5. Give a counterexample to this statement: Using only letters, numbers, and underscores will always form a valid variable name.
8. Let’s say a piece of code does not work as expected. How do you determine whether it is a syntax error or a logic error?
In-class Exercise (Exam 1 Review Worksheet)

How many times is “hi” printed? What is the value of \( i \) and \( j \) after the program terminates?

\[
i = 23 \\
\text{while}(i > 12): \\
\quad j = i - 2 \\
\quad \text{while}(j < 30): \\
\quad \quad \text{print}("hi") \\
\quad \quad j = j + 10 \\
\quad i = i - 3
\]
Exam 1 Review Worksheet

1. Logic gates is a level of abstraction between microarchitecture and transistors. How exactly does logic gates connect microarchitecture to transistors?
Exam 1 Review Worksheet

2. Recall the layers of abstraction involved in computing, from the very highest system level all the way down to the transistor level. What layer of abstraction is described in each of the following statements?

2.1 A bunch of loads into memory, stores from memory, arithmetic computational instructions, and a branching instruction at the ISA level, is all contained in one single Python "if" statement.

2.2 This contains both the architecture of a computer as well as the instruction set for each different component.

2.3 An ALU consists of many AND and OR gates.

2.4 A very common type of 2-input NAND gate consists of 4 transistors.
Exam 1 Review Worksheet

3. What role does the number sign, pound sign, or hashtag character (#) play in Python? Give an example of using it.
10. What is an algorithm? What are trade-offs when it comes to writing an algorithm?
Exam 1 Review Worksheet

11. What is a computer? What is the difference between a stored-program device and a fixed-program device?