*CS 101 – Review for final exam*

Outline of concepts

1. Key terms: computer, computer science, hardware, software, analog, digital
2. Historical milestones
3. Binary numbers, conversions, octal, hex  
   a. Convert between decimal and binary  
   b. Familiarity with powers of 2  
   c. The number of possible representations for n bits  
   d. Octal and hexadecimal shorthand for binary
4. All information is stored in binary; software and file type determine the precise representation and how translation is done; size and scheme
5. Integer representations: unsigned, signed, biased  
   a. Range of values for each representation  
   b. How to represent negative numbers
6. Real number representation  
   a. Extension of the place value system to negative powers  
   b. What happens to a binary representation when you double or halve a value  
   c. How to convert a real number to binary. The representation may contain a repeating pattern.  
   d. Scientific notation: decimal versus binary, normalized  
   e. Role of sign, exponent, mantissa   
   f. Distribution of values, holes in the representation, ramifications of the choice of number of bits of exponent and mantissa
7. Text representation: properties of ASCII and Unicode
8. Data compression techniques: keyword encoding, run length encoding, Huffman code  
   a. Given the distribution of symbols, derive a Huffman code; Know how to encode and decode according to a Huffman code
9. Image representation   
   a. terminology (pixel, landscape, portrait, aspect ratio, resolution, dynamic range, binary image, pixelation, sampling, quantizing, dithering)  
   b. schemes: RGB, Indexed color
10. Logic gates:   
    a. How AND, OR, XOR, NOT, NAND and NOR gates work, truth table  
    b. digital circuits built from several gates such as an adder
11. CPU instruction execution
12. Memory: classification as main or secondary, memory hierarchy, media, examples, disk terminology (seek time, latency, transfer)
13. OS role and responsibilities (namely: passwords, CPU mgmt., memory mgmt., file mgmt., I/O, user interface)
14. CPU Scheduling, system load  
    a. Scheduling algorithms: First come first served, Shortest job next
15. File permissions, file organization,
16. The nature of problem solving, software; problem-solving process
17. Software elements: program, types of statements, algorithm, structure of solution (e.g. loop), I/O, examples, 3 kinds of errors
18. Programming languages: machine, assembly, high-level
19. Searching and sorting  
    a. 2 search techniques if the data is arranged linearly: linear search, binary search  
    c. 4 sorting algorithms: insertion, selection, merge, bubble
20. Steganography
    1. Purpose
    2. various techniques such as pinprick, microdotting, modifying pixels
21. Cryptography terminology: plaintext, ciphertext, cipher system, key, cryptanalysis
22. Types of cipher systems: transposition vs. substitution
    1. Caesar cipher vs. general substitution cipher
    2. Improvements to substitution cipher: Null, nomenclator; homophonic and polyalphabetic ciphers
    3. Great cipher; book cipher; Vigenère cipher; one-time pad; JN-25; Enigma
    4. How many possible keys? How to choose a key; how to distribute a key
23. Breaking codes
    1. Babbage able to break a Vigenère cipher
    2. extra information from patterns of radio transmission
    3. Polish and British codebreakers exploit weaknesses in Enigma
24. Diffie-Hellman key exchange protocol (basic ideas)
25. RSA (basic ideas): asymmetric cipher
26. How to calculate the modulo of a large power; one-way function
27. Digital signature and hash function
28. Quantum computer and quantum cryptography

Practice Questions

1. Give the general name of a program that translates high-level computer source code into binary code.
2. Some programming languages are machine dependent, while others are machine independent.
   1. What does it mean for a programming language to be machine dependent?
   2. When designing software, which type of language is generally preferred, a machine dependent or machine independent one? Why?

* 1. Give one example of each type of language.

1. Explain how a computer program can be written in such a way that sometimes it runs correctly, but other times it emits a run-time error, depending on the input data.
2. Suppose that L is the name of a list of ten numbers in a Python program. Write an expression that finds the sum of the first number and last number in the list.
3. What is wrong with the following Python code that attempts to find the smallest number in a list? Show how you would fix it.

min = 0

for number in L:

if number < min:

min = number

1. The following Python program attempts convert a time in Denver to a time in Atlanta by adding 2 hours. However, it does not work in some cases. Show how you would fix the program.

hour = input(“Enter Denver hour”)

min = input(“Enter Denver minute”)

hour = hour + 2

if min < 10:

print “The time in Atlanta is “ + str(hour) + “:0” + str(min)

else:

print “The time in Atlanta is “ + str(hour) + “:” + str(min)

1. Describe in English an algorithm for determining how many times the letter T (capital or lowercase) appears in a sentence.
2. How could we design a program to count the number of words in a sentence?
3. Suppose a list of numbers [ 5, 4, 7, 3, 2 6 ] is going to be sorted in ascending order using the following steps. Name the sorting method that is being used.

5 4 7 3 2 6

4 5 7 3 2 6

4 5 7 3 2 6

3 4 5 7 2 6

2 3 4 5 7 6

2 3 4 5 6 7

1. Repeat the previous question: we are going to sort the list again in ascending order. Name the sorting method that is being used.

5 4 7 3 2 6

4 5 7 3 2 6

4 5 7 3 2 6

4 5 3 7 2 6

4 5 3 2 7 6

4 5 3 2 6 7

4 3 5 2 6 7

4 3 2 5 6 7

4 3 2 5 6 7

4 3 2 5 6 7

3 4 2 5 6 7

3 2 4 5 6 7

3 2 4 5 6 7

3 2 4 5 6 7

3 2 4 5 6 7

2 3 4 5 6 7

1. Write down a bit string having 6 bits. What number does this bit string represent in the following representation schemes?
   1. unsigned
   2. signed
   3. biased-32

Practice this question with a bit string that starts with a 1, and a bit string that starts with a 0.

1. Pick a two digit number (integer). Determine the 8-bit representations for this number using the following representation schemes
   1. unsigned
   2. signed
   3. biased-127

Practice this question also with a two digit negative number.

1. The single-precision floating-point representation is a method of representing real numbers. We allocate 8 bits for the exponent.
   1. Explain how we can determine the approximate bound for the largest possible (finite) number that we can represent, and the smallest positve number.
   2. How would your answers change if we had instead allocated 7 bits for the exponent? Would this be practical for scientific applications?

1. This question features some approximate values for large powers of 2. These approximations will appear in brackets. You may find the question easier to grasp initially in terms of these approximate values than just the powers of 2.   
   Using the fact that the mantissa for a single-precision floating-point number has 23 bits:
   1. How many real numbers can be exactly represented between two consecutive powers of 2?
   2. Consider the real numbers that we can exactly represent between 8 and 16. How far apart are they?
   3. How far apart are the exact real numbers between 2^23 [8 million] and 2^24 [16 million]?
   4. How far apart are the exact real numbers between 2^24 [16 million] and 2^25 [32 million]?
   5. What does the answer to the previous question tell us about real number versus integer representation?
   6. Explain why a financial account balance should not be stored in this format. Illustrate the problem with a specific example.

1. The binary number 110.1001 equals what base-10 number? If we multiply this binary number by 4, what is the binary result?

1. In what ways did typical computers of the 1980s differ from those of the 1950s? How do today's PCs differ from those of 20 years ago?

1. If the ASCII code for 'A' is 65, what symbol has ASCII code 85?

1. The ASCII codes for 'a' and '0' (zero) are 97 and 48, respectively. What is the ASCII code for '3' ? What is the ASCII code for '6' ? The value '3' + '3' is the ASCII code for what letter? When you ask the computer for 3+3, how come it knows to give you 6 as the answer instead of a letter?

1. Basic image properties:
   1. What does an image look like if we reduce its dynamic range too much? What if we reduce its resolution too much?
   2. How much memory does an image need if it uses a grayscale system with 16 possible shades of gray, and it measures 400 by 300 pixels?
   3. What are the dimensions of your computer's screen?

1. Give example RGB codes for light green, medium green and dark green.
2. When we use indexed color, we often express the R,G and B values in hexadecimal, and the values we can choose are 00, 33, 66, 99, cc and ff. What color is represented by the code #cc0000?

1. Using indexed color and dithering, explain how we can approximate a background color close to the value #993355.

1. Suppose an XOR gate has two inputs x and y and an output z. If x = 1 and z = 1, is it possible for us to determine the value of y? If so, what is it?
   1. Practice this question with other logic gates, and other values of x and z.

1. Consider a NAND gate with two inputs x and y and an output z. If x = y, then what can we conclude about z?

1. Suppose a machine does 8-bit signed integer arithmetic. Give an example of a computation that will result in overflow. Is underflow possible?
2. The contents of sixteen bytes of data can be described by how many hexadecimal digits?

1. Explain how the speed of light relates to how fast information can be processed in the CPU, and how fast data can be transmitted over a long distance network. In other words, which orders of magnitude are appropriate?

1. The problem-solving procedure typically involves 5 steps. Identify an obstacle that could occur at each step.

1. Suppose you are trying to guess a number between 1 and 100 inclusive. Using the binary search strategy, list the guesses you would make if the correct answer is 70. You may assume that after each wrong guess you would be told whether to go higher or lower.

1. Given this list of values: [ 9, 2, 5, 4, 8, 3 ], show the steps we perform to sort this list in ascending order if we use each of the following sorting techniques.  
   a. bubble  
   b. selection  
   c. insertion  
   d. merge

1. Suppose a computer's hard drive contains 10,000 files. Explain how these files can be organized into folders in such a way that no folder contains more than 20 folders or files.
2. Explain what the following octal permission codes mean. Which ones would be appropriate only for folders and executable files?  
   a. 770  
   b. 644  
   c. 400  
   d. 754

1. Suppose a book contains 700 pages. On each page there are typically 500 words of text and an image measuring 600 by 400 pixels and is in full RGB color. In theory, how much memory would be required to store the contents of this book?
2. Explain why a homophonic cipher is more difficult to break than an ordinary substitution cipher.
3. Compare the Vigenère cipher with the Caesar cipher.
4. Compare the Pigpen cipher with the ordinary substitution cipher.
5. Compare a one-time pad with a Vigenère cipher.
6. How large of a text message can be hidden inside a 200x200 RGB image? Assume the image depicts something busy with much detail such as a tropical rain forest.
7. In quantum cryptography, which photons of light pass through a Polaroid filter that is oriented vertically?
8. If we use the XOR function to encrypt a message, how do we decrypt the ciphertext? What is the key in this case?

CS 101 Answers to Final Exam Review Practice Questions

1. Compiler
   1. Machine dependent means the language is suited only for one type of machine (Intel, Mac, AMD, Sun, etc.)
   2. Machine independent is preferred. We would like to have programs that can run on any machine.
   3. Example of machine dependent: any machine’s assembly language. Example of machine independent: Python.
2. (Many possible correct answers.) One statement in the program may divide by a variable. If the value of this variableis zero, you have a run-time error.
3. L[0] + L[9]
4. If all the numbers in the list are positive, the program will report 0 as the smallest number, even though 0 is not in the list. Change the first statement to: min = L[0]
5. Immediately after the statement “hour = hour + 2”, add the following:

if hour > 12:

hour = hour – 12

1. count = 0

for each letter in the sentence

if the letter is capital or lowercase T

count = count + 1

1. A simple way to estimate the number of words in the sentence is to count the spaces, much like we counted the T’s in the previous question. To make the solution more general, we would have to ignore multiple consecutive spaces. For example, if someone types two spaces after a colon, we want to count that as just one space.
2. Insertion sort
3. Bubble sort
4. Let’s try the bit string 100110. This is the binary code for:
   1. 32+4+2 = 38 in unsigned
   2. 38 – 64 = –26 in signed
   3. 38 – 32 = 6 in biased-32

Now let’s look at a bit string that starts with 0. Consider 010101. This is the binary code for:

1. 16+4+1 = 21 in unsigned
2. 21 in signed (same as unsigned)
3. 21 – 32 = –11 in biased-32
4. Let’s consider the number 59. If we write this number as a sum of powers of 2 (which we would get on our receipt if we shop at the binary store), we have 59 = 32+16+8+2+1.
   1. Unsigned = 00111011
   2. Signed = 00111011 (same as unsigned)
   3. Biased-127 = same as unsigned representation of 59+127 = 186 = 128+32+16+8+2 = 10111010.

Now let’s try –59.

1. Unsigned is impossible.
2. Signed = 11000101
3. Biased-127 = same as unsigned representation of –59+127 = 68 = 64+4 = 01000100.
   1. If we have 8 bits for the exponent, then we have about 2^8 = 256 possible values for the exponent. These exponents should be evenly divided between positive and negative exponents. Thus, the largest power of 2 we can use is about 2^128. It’s more convenient to work in base 10, and since log 2 is about .301, we have 2^128 ~= 10^38. This is an approximate maximum value. Similarly, the smallest positive negative number will be the negative exponent: 10^–38.
   2. If we had instead used 7 bits for the exponents, then we would start with about 128 possible exponents, with the largest possible exponent value about 64. Then 2^64 is about 10^19. This most minuscule positive number would analogously be about   
      10^–19. The question asks us to consider if this would be practical for science. It would not, because in many applications in chemistry and physics we need to express values with larger exponents than 19 (and less than -19). Some famous examples are: Avogadro’s number which uses 10^23, and the mass of an electron which is on the order of 10^–31 kg.
4. Let’s continue to use the approximation that 2^20 = 1 million.
   1. 2^23 = 8 million
   2. The distance from 8 to 16 is 16 – 8 = 8. The 8 million numbers are spread out over the range of real numbers from 8 to 16, so the density is 8 million / 8 = 1 million per integer. Thus, the individual real numbers are 10^–6 (i.e. one one-millionth) apart.
   3. The distance from 8 million to 16 million is 8 million. The 8 million real numbers are spread out over a distance of 8 million, so the density is 8 million / 8 million = 1. Thus, the individual real numbers are 1 integer apart.
   4. This part is just like the previous two with different numbers plugged in. The density of the real numbers is (how many reals) / (distance between the extremes) = 8 million / 16 million. Thus, there is 1 real number for every 2 integers, so they are 2 units apart. In other words, the real numbers are restricted to being just the even integers. Odd integers now cannot be exactly represented.
   5. As a result of our calculations above, once you go past about 16 million, you begin to skip some integer values. In the comparable 32 bit integer representation schemes, they are able to represent consecutive values into the billions.
   6. Because the mantissa has 23 bits, we can only guarantee precision to one part in 8 million. This is about 7 significant figures of precision. For financial calculations, you need to be correct to the nearest .01. Thus, if you have an amount of money that is 7 orders of magnitude higher than .01 (which is 10^5 dollars or $100,000), you would not be able to represent it exactly. To make matters worse, every time you perform a real-number calculation, there is likely a tiny error present, and these errors can accumulate. As a result, it’s more realistic to use double precision instead of single precision. Another solution is to store the dollars and cents as separate integer values to avoid any trouble with precision.

1. 4 + 2 + (1/2) + (1/16) = 6-9/16 (i.e. six and nine-sixteenths) or 9.5625. If you multiply by 4, note that 4 = 2^2, so move the “.” by 2 places to the right to obtain 11010.01.
2. 1980s computers were much smaller, faster and cheaper than those of the 1950s, to the point that it was feasible for many households and small businesses to own one. Today’s machines have continued the trend. Increasingly, the machines we use today are small enough to carry around routinely. Also since the 1980s is the need for machines to be part of a network.
3. ‘A’ is the first letter of the alphabet. If we add 20 to its ASCII code, we want the 21st letter, which is ‘U’.
4. The ASCII code for ‘3’ is ‘0’ + 3 = 48+3 = 51. Similarly, the ASCII code for ‘6’ is ‘0’ + 6 = 48+6 = 54. When we add ‘3’ + ‘3’ we obtain 51+51 = 102. If ‘a’ = 97 then we want the letter which is (102 – 97 =) 5 letters after ‘a’, which is ‘a’ + 5 = ‘f’. When we ask the computer to add 3+3, the machine can distinguish between numbers and text. It can tell we are entering numbers. But the machine needs to be told in advance to expect numerical rather than text input. For example, in Python we have distinct functions input() and raw\_input().
5. 1. Reducing the dynamic range too much will create a binary image, or an image where we see false stripes where we can see the boundary between two sharply contrasting colors. If we reduce the resolution too much, the pixels become large enough to see, and we say the image is pixelated.
   2. If we want 16 shades of gray, we need 4 bits or ½ a byte per pixel. The image measures 400x300, so there are 120,000 pixels. Thus, the image size is about 60 KB.
   3. (many possible answers) 1440x900 is a typical monitor resolution. You can find this out by right clicking on your computer’s desktop background, and select the “Settings” tab.
6. Medium green would be (0, 255, 0). To obtain light green, we need to increase the red and blue numbers the same amount, so for example (80, 255, 80). To create a dark green from medium green we can reduce the green value, for example (0, 150, 0).
7. If we had #ff0000 this would represent (255, 0, 0) which is medium red. Since we have a red number of cc instead of ff, this represents a somewhat darker red.
8. There is no “55” value for indexed color, but this is between “33” and “66”. This means that in order to emulate the color #993355, we need to perform dithering on these two pixel values: #993333 and #993366. Since 55 is twice as close to 66 as it is to 33, we would use twice as many #993366 pixels as #993333 pixels.
9. An XOR gate outputs 1 if its two inputs are different. In our case, we know that z=1, so x and y must differ. Since x = 1, we see that y = 0.
   1. As another example, consider an AND gate instead of XOR. If the output of an AND is 1, this means that both inputs had to be 1, so y = 1. Finally, let’s consider an OR gate. If its output is 1, we don’t know much about the inputs except that they cannot both be zero. Knowing that x = 1 doesn’t help, because both y = 0 and y = 1 are plausible.
10. NAND is the opposite of AND. If the two inputs of an AND gate differ, this means one of the two inputs is a 0, so the output must be 0. If AND outputs 0, then NAND outputs 1.
11. Note that the range of possible values is –128 to +127. We just need to pose a question whose correct answer is outside this range. For example, 100+100. On the other hand, underflow is only possible in real-number arithmetic, not integer arithmetic.
12. A hexadecimal digit is equivalent to 4 bits or ½ a byte. Therefore we need two hexadecimal digits per byte. If there are 16 bytes, we need 16x2 = 32 hex digits.
13. The speed of light is about 1 foot per nanosecond. To think about large distances, let’s multiply by a million. 1 million feet is about 200 miles, and 1 million ns = 1 millisecond. So we can say the speed of light is about 200 miles per millisecond. Therefore, when inside the CPU it makes sense to use time units of nanoseconds, but for the Internet flight times (the time to send the shortest possible message) we would use milliseconds.
14. Problem solving obstacles at each step:
    1. We may not understand the problem.
    2. We may understand the problem but not know how to solve it.
    3. We may understand how to solve the program, but we don’t know how exactly to express our steps in the programming language.
    4. After we type in our program on the computer, we may have a syntax error, so that it cannot run.
    5. As we run the program, we may obtain incorrect output because of a logical error.
15. Our first guess is 50.   
    We need to go higher, so our second guess is (50+100)/2 = 75.   
    Next, we need to go lower, so our third guess is (50+75)/2 = 62.  
    Next, we need to go higher, so our fourth guess is (62+75)/2 = 68.  
    Next, we need to go higher, so our fifth guess is (68+75)/2 = 71.  
    Next, we need to go lower, so our sixth guess is (68+71)/2 = 69.  
    Finally, we need to go higher one last time to obtain (69+71)/2 = 70.

1. We need to sort the list of values in ascending order: [ 9, 2, 5, 4, 8, 3 ].  
   a. Here is bubble sort:  
   9 2 5 4 8 3 is the starting order  
   2 9 5 4 8 3  
   2 5 9 4 8 3  
   2 5 4 9 8 3  
   2 5 4 8 9 3  
   2 5 4 8 3 9 now the 9 is in the right spot  
   2 5 4 8 3 9  
   2 4 5 8 3 9  
   2 4 5 8 3 9  
   2 4 5 3 8 9 now the 8 is in the right spot  
   2 4 5 3 8 9  
   2 4 5 3 8 9  
   2 4 3 5 8 9 now the 5 is in the right spot  
   2 4 3 5 8 9  
   2 3 4 5 8 9 now the 4 is in the right spot, and we are actually done.   
     
     
   b. Here is selection sort:  
   9 2 5 4 8 3 is the starting order  
   2 9 5 4 8 3  
   2 3 5 4 8 9  
   2 3 4 5 8 9 and at this point we are done. The next two steps don’t change the list.  
     
   c. insertion  
   9 2 5 4 8 3 is the starting order  
   2 9 5 4 8 3  
   2 5 9 4 8 3  
   2 4 5 9 8 3  
   2 4 5 8 9 3  
   2 3 4 5 8 9  
     
   d. merge   
   9 2 5 4 8 3 is the starting order  
   Split into two lists: 9 2 5 and 4 8 3.  
   Split each list again, so that we have two pairs of lists:  
   9 2 5 4 8 3  
   Within each list, we sort in ascending order by swapping if necessary.  
   9 2 5 4 3 8  
   Now we collate the first two lists, and the second two lists to re-create lists of length 3.  
   2 5 9 3 4 8  
   Finally, we collate these two lists into our entire list of length 6:  
   2 3 4 5 8 9
2. We need to organize 10,000 files into folders so that no folder has more than 20 folders or files. To do this, we need to have a hierarchy of folders within folders. Here is one way to do it. Your hard drive can have 20 folders in it, and let’s say they are named A through T. Each of these 20 folders will themselves contain 20 folders, and let’s say they are named 1 through 20. At this point, the bottom of your file system will have 400 folders, with names like A\1, A\2, A\3, … A\19, A\20, B\1, B\2, … T\20. If we place our files into these folders evenly, we would average 25 files per folder. Unfortunately, the problem said we can only have 20 files per folder, so this means that some of these folders need to have subfolders themselves.
3. Octal permission codes. Note that folders and executable files are the only types of objects where we would want execute permission. In other words, anytime you see an odd digit, this implies execute permission. So, the codes 770 and 754 (the codes containing at least 1 odd digit) would be appropriate for folders and executable files, while 644 and 400 (containing only even digits) would be appropriate for general files.  
   a. 770 = The owner and anyone in the owner’s group may do anything.  
   b. 644 = The owner may read or write, but everyone else may only read.  
   c. 400 = The owner has read-only acces, while everyone else has no access.  
   d. 754 = The owner may do anything, members of the owner’s group have read and execute permission, and everyone else just has read access.
4. A typical word is 5 letters long and is followed by a space or punctuation. So, let’s say we need 6 bytes per word. If there are 500 words on a page, we would need 6x500 = about 3 KB to store the words of a page. An image measuring 600x400 pixels has 240,000 pixels. If we use CMY color, this uses 3 bytes per pixel, so we would need 720,000 bytes per page just for the image. We see that the amount of memory for the text is tiny relative to the image! If there are 700 pages, the total memory requirement would be about 700 x 723 KB = about 504 MB. In retrospect, most books would not require this much space, because we don’t have so many (large) images, and we can compress them.
5. In a homophonic cipher, the more common letters such as E and S are allocated several different codes. Typically the resulting ciphertext consists of two digit numbers from 00 through 99. Each value appears approximately 1% of the time. The frequency distribution is almost perfectly flat. But in an ordinary substitution cipher we can see right away which symbols are the most common, because some letters are used up to 10% of the time.
6. A Vigenère cipher is essentially many Caesar ciphers at once, so it’s harder to crack. Each letter of our message uses a different Caesar cipher key, up to the length of the Vigenère key. The Vigenère cipher is tougher to crack because the cryptanalyst does not know the length of the key. Once the key length is known, then each Caesar cipher key must then be determined.
7. A Pigpen cipher is an ordinary substitution cipher. It’s just that instead of substituting plaintext letters with ciphertext letters, the ciphertext consists of geometric symbols.
8. A one-time pad is a type of Vigenère cipher: in this case we have the best possible key, one that is both long and randomly generated.
9. There is no unique answer to this question, but here is a plausible approach. We can change the green value of several pixels. How many pixels can we afford to modify? Let’s assume that the ASCII codes of our letters are not much different from the pixel values of the original image. In other words, changing pixels won’t be too obvious. If we change 1 pixel per row of the image by setting its green number to the ASCII value of one character, then we can insert 200 bytes into the image. We can attempt to be more aggressive, by adding characters to 10% of the pixels of the image, which would insert 4000 bytes. We probably don’t want to squeeze much more than this into the picture.
10. All of the light that is oriented vertically, half of the light that is oriented diagonally (both / and \), and none of the light that is oriented horizontally.
11. If we use an XOR cipher, the encryption and decryption keys are the same. In fact, the process of encrypting and decrypting is the same. XOR is its own inverse operation. (Just like if you multiplied a number by –1.)