CS 105 Review Questions

Most of these questions appeared on past exams.

1. What is the minimum number of bits needed to store a single piece of data representing:
   a. An integer between 0 and 100?

   b. An integer between 1600 and 2000?

   c. A U.S. state?

   d. One of the 200 billion stars in the Milky Way galaxy?

   e. A Chinese character, assuming that 10,000 different characters exist?

   f. A room in a large hotel? (Please indicate how many rooms you are assuming.)

2. Convert these hexadecimal numbers into binary and octal.
   a. BEEF

   b. 7c

   c. ff

   d. 33ff99

3. Convert the following values to binary and hexadecimal.
   a. $(13 \times 16^3) + (7 \times 16^2) + (9 \times 16^1) + (12 \times 16^0)$

   b. $2^{16} + 2^{18}$

4. Convert the following into hexadecimal:
   a. The octal number 07654.

   b. The binary number 11100011110.

5. Convert the following into octal and hexadecimal:
   a. The binary number 1110 0111 1010 0110.

   b. The decimal number 87.

6. The highest digit in hexadecimal is _____, and its value is _____.

7. The contents of sixteen bytes of data can be described by how many hexadecimal digits?

8. Suppose a binary number needs 7 bits to be represented. In other words, 6 bits would not be enough and 8 bits are not necessary. If this number is written in hexadecimal, what is the first (leftmost) digit?

9. Suppose x is a 30-bit binary number, and y is a 32-bit binary number. If the first 30 bits of y are the same as x, and the last two bits of y are 11, then what is the relationship between the values of x and y?

10. 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.

11. Describe in English an algorithm for determining how many times the letter T (capital or lowercase) appears in a sentence.

12. How could we design a program to estimate the number of words in a sentence?

13. What is the difference between a computer program and an algorithm? Which one is generally created first in the problem-solving procedure?

14. Define the following terms:
    a. Iteration
    b. Algorithm
    c. Logical error
    d. Software
    e. Variable

15. A simple program usually consists of three phases: _____, followed by _____, followed by _____.

16. Describe the steps necessary in the computer problem-solving procedure. Identify an obstacle that could occur at each step.

17. Write a Pascal statement that will subtract 5 from the variable count and put the result in a new variable called adjusted_count.

18. Suppose w is a string variable containing a word.
    a. Explain how you would determine whether the letter 'z' appears in w. A Pascal function might help.

    b. How would your answer to the previous question change if we wanted to know the number of times the letter 'z' appears?

    c. What if we wanted to determine the location of the first 'z' in w?

19. Show in pseudocode how you would get the computer to print the numbers from 1 to 26, using a loop. How would you change your answer if we wanted to print the numbers in reverse order?

20. Consider the following Pascal code:

```
if _____ then
        valid := true
else
        valid := false;
```

    Notice that the above code will set the variable valid to true if some condition (as yet unspecified) satisfies a certain condition. For each of the following cases, indicate what condition we should enter in the blank in order for the variable valid to be set to true. In all cases, assume that the number entered is an integer.

    a. The number is a positive 3-digit number.

    b. The number is between 18 and 65 inclusive.

    c. The number is <u>not</u> between 18 and 65 inclusive.

    d. The number is positive.

e. The number is a multiple of 5, but not a multiple of 4.

f. The number is an odd number.

g. The number is 7, 11 or 13.

h. The number is <u>not</u> 7, 11 or 13.

21. Given a number of cents (anywhere from 1 to 99), write an algorithm to determine the appropriate change, i.e. the number of quarters, dimes, nickels, and pennies.

22. In Pascal, what do the characters *) and (* mean? What is their purpose?

23. What is the output of this Pascal code?

```
for i := 2 to 7 do
        writeln (i);
```

24. What are the values of these Pascal expressions?
    a. 14 / 5

    b. 14 mod 5

25. What is the output as the following Pascal loop executes?

```
for i := 1 to 10 do
    writeln (22 – 2 * i);
```

26. What type of computer error results when we ask the computer to perform an illegal operation such as taking the square root of a negative number, causing the program to immediately halt?

27. Describe in English how we would design a computer program that solves this problem: Ask the user for the lengths of the three sides of a triangle, and determine if it is isosceles.

28. Suppose we want to know if the Pascal variable num is divisible by either 7 or 11. The following if-statement's condition has been written incorrectly. Show what the mistake(s) is/are, and show the correct form.

```
if num mod 7 or 11 := 0 then …
```

29. The computer problem-solving procedure can be divided into five steps. They are given below, but they are out of order. Indicate the correct order of these steps:

    _____

    a. The computer program is translated into machine language
    b. We describe how to solve the program in English
    c. Test the program to see if it gives the correct output
    d. Type the program into the computer.
    e. Make sure we understand the problem specification.

30. A leap year occurs when the year number (e.g. 1984) is divisible by 4. But there is a special case for years ending in 00: these must be divisible by 400 to be considered a leap year. Thus, 1900 was not a leap year, but 2000 was. Write an if-statement that determines if the integer variable year represents a leap year. Hint: use the mod operator for taking remainders.

31. The following code attempts to count how many times the capital letter R appears in a string called word. But there is a mistake in the code. Why will the code not work as is? How should the mistake be corrected?

```
for i := 1 to length(word) do
  begin
     count := 0;
     if word[i] = 'R' then
          count = count + 1;
  end;
writlen(count);
```

32. What should be written at the beginning of a computer program in order to assist a person who may want to read and understand your program?
    a. Assignment statement
    b. Comment
    c. Input statement
    d. Output statement

33. An algorithm is _____.
    a. a problem that we wish to solve using the computer
    b. the solution to a problem that is solved by computer
    c. a language used for writing instructions
    d. the set of all mathematical operations available on a computer

34. In Pascal, what keyword introduces a block of code that is meant to be executed repeatedly?

35. The following Pascal 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.

```
write('Enter Denver hour:  ');
readln(hour);
write('Enter Denver minute:  ');
readln(minute);
hour := hour + 2;
if minute < 10 then
        writeln('The time in Atlanta is ', hour, ':0', minute)
else:
        writeln('The time in Atlanta is ', hour, ':', minute);
```

36. Define each of the following types of programming errors, and describe the behavior the computer would exhibit to indicates to us that our program has this kind of error.
    a. Syntax error

    b. Logical error

    c. Run-time error

37. When someone is designing software, what step should take place after the algorithm has been discovered?
    a. compile the program to check for errors
    b. pipeline the operations to make the program run faster
    c. write the program in a programming language
    d. run the program to make sure it gives the correct output
    e. fetch the next instruction from memory
    f. save the program to disk

38. What is the value of the variable sum after the following code executes?

```
sum := 0;
number := 1;
while number <= 10 do
  begin
    sum := sum + number;
    number := number + 3;
  end;
```

39. Suppose a computer program contains an instruction to open a file for reading. However, this file does not exist. What sort of error does this represent?

40. Suppose you just finished typing in a computer program. But one of the lines that reads "if x > 0" should instead be "if x < 0", because you had meant to look for a negative number. What type of error is this?

41. What is the difference between an algorithm and a computer program?

42. Complete the following if-statement so that it checks to see if the integer value n is between 1 and 12 inclusive.

```
if _____________________________________ then
```

43. Suppose s is a string variable in a Pascal program. Let's determine if the last two characters in s are the same. Fill in the blank for this if-statement.

```
if _____________________________________ then
```

44. Suppose that n is a variable in a Pascal program containing a positive integer. Let's determine if this value ends in the digits 65. For example, numbers such as 265, 1965, 99965, etc. Fill in the blank for the if-statement.

```
if _____________________________________ then
```

45. If we were to write out the number 25 million into binary, how many bits would be necessary?

46. Convert the decimal number 44 into binary.

47. Convert the binary number 1101110 into decimal, octal, and hexadecimal.

48. Consider the following Pascal code. What is the output when it runs?

```
n := 0;
while n <= 12 do
  begin
    writeln(n);
    n = n + 3;
  end;
```

49. Suppose that s is a string variable. Using Pascal code or pseudo-code, show how we can determine how many times the capital letter 'A' appears in s.

50. For each of the following examples of computer errors, classify it as logical, run-time, or syntax error.
    a. Writing an assignment statement backwards, as in 3 := c

    b. Unnecessarily putting quotation marks around integers when you want to add them, as in concatenating the strings '8' + '4' resulting in '84' when instead you wanted 8 + 4 = 12.

    c. Forgetting the closing right parenthesis ')' when using writeln.

    d. Asking if the value of n mod 6 equals 8.

    e. Printing the numbers from 1 to 11, when you had intended to print 1 to 10.

    f. Forgetting to put a (* symbol before the start of a comment.

51. What is the simplified value of this Pascal expression?
    7 div 3 – 1 + 17 mod 5 * 3

52. Suppose that s is a string variable in Pascal. This string contains: 'Silicon Gold'. Note that the string does not include the quotes but does include a space between the two words. Indicate the value of the following expressions:
    a. s[5]

    b. s[length(s) – 3]

53. What does the following Pascal code accomplish? Assume that s contains a string. Note that pos(c, s) determines the index location within s where we find the first occurrence of c. If c does not exist in s, pos returns zero.

```
count := 0;
for i := 1 to length(s) do
    if pos(s[i], 'xyz') > 0 then
        count := count + 1;
writeln(count);
```

54. What is the output of the following Pascal code? What general mathematical problem is this code solving?

```
n := 12;
d := 1;
while d < n do
  begin
    if n mod d = 0 then
      writeln(d);
    d = d + 1;
  end;
```

Answers to review questions.

1. Minimum number of bits…
    a. 7
    b. 9
    c. 6
    d. 38
    e. 14
    f. In a 300-room hotel, for example, we would need 9 bits.

2. Converting to binary and octal:

a. Binary = 1011 1110 1110 1111
   To convert to octal, we need to regroup the binary into groups of 3:
   001 011 111 011 101 111 = 137357 in octal
b. Binary = 0111 1100
   Regrouping in threes, the binary looks like 001 111 100 = 174 in octal
c. Binary = 1111 1111
   Regrouping in threes, the binary looks like 011 111 111 = 377 in octal
d. Binary = 0011 0011 1111 1111 1001 1001
   Regrouping in threes, the binary looks like 001 100 111 111 111 110 011 001 which equals 14777631 in octal.

3. Converting to binary and hexadecimal:
   a. The hexadecimal is d79c, and the binary is 1101 0111 1001 1100.
   b. The hexadecimal is 50000, and the binary is 0101 0000 0000 0000 0000.

4. Converting to hexadecimal:
   a. Octal 7654. First, convert to binary: 111 110 101 100. Regroup in groups of 4: 1111 1010 1100, which equals fac in hexadecimal.
   b. This binary number should be grouped in fours working from the right. We have: 0111 0001 1110 = 71e in hexadecimal.

5. Converting to octal and hexadecimal:
   a. Binary 1110 0111 1010 0110 can immediately be converted to hexadecimal as e7a6. To convert to octal, regroup in threes: 001 110 011 110 100 110 = 163646 in octal.
   b. Let's go to the binary store to convert 87 to binary. The powers of two that add up to 87 are: 64 + 16 + 4 + 2 + 1. Thus, the binary representation is 0101 0111. This is equivalent to 57 in hexadecimal. Regroup in threes, this binary number will look like 001 010 111 = 127 in octal.

6. F, 15

7. Each hexadecimal digit is 4 bits or ½ of a byte. Thus, 16 bytes is equivalent to 32 hex digits.

8. If exactly 7 bits are required, then we know the 7^{th} bit from the right is 1, and bits farther to the left would have been all zero. So, the number is of this binary form: 01xx xxxx, where each x can be either a 0 or a 1. The question asks about the first hex digit, which could be represented in binary by anything of the form 01xx, which is 0100 through 0111, inclusive. These are the digits 4, 5, 6 and 7.

9. To obtain y from x, we shift x by 2 bit positions to the left and then add 3. Each time you shift left by 1 position you are doubling the number. So, $y = 4x + 3$.

10. For example, we might have the user enter 2 numbers, and output their quotient. If the second number entered is 0, we would have a run-time error resulting from division by zero.

11. First, set count equal to 0.
    For each letter in the string that contains the sentence,
      If this letter is a T or t,
        then increment count by 1.
    Output the count.

12. We could assume that the sentence is formatted so that there is a space between each word. Then, we can traverse the characters of the string, counting the spaces. Add 1 to this number, and we now have the number of words.

13. The essential difference is that an algorithm is stated in English, while a computer program must be typed out in a computer programming language.

14. Definitions:

    a. An iteration is one pass through the body of a loop.
    b. An algorithm is a list of steps written in English (or whatever your native or fluent language is) that describes how to solve a problem. This set of steps must be unambiguous, detailed and deterministic. (In this case, deterministic means that at any point in the algorithm we know what the next step should be, and the algorithm must terminate.)
    c. A logical error is a mistake in a program that occurs when the output is incorrect, yet the computer did not itself tell you that you have an error.
    d. Software consists of the computer programs that run on your machine.
    e. A variable is a place in memory used to store a value used in a computer program. For our purposes, a variable has a name and contains a value.

15. Input, calculations, output

16. First, read and understand the problem. The obstacle may be that you do not understand what the problem is asking.
    Second, write an algorithm. Here, you might not know how to solve the problem.
    Third, transcribe your solution into a programming language such as Pascal. Here, the problem may be that you do not know enough about the programming language to convey what you want the computer to do.
    Fourth, compile the program. The obstacle here is that the program may contain a syntax error.
    Finally, run and test the program. The problem here may be that the program aborts while executing due to a run-time error, or that the output from the program is incorrect.

17. adjusted_count := count – 5;

18. In this question, the variable w refers to a string containing a word.
    a. Thanks to the Pascal string function pos, we don't need a loop. Using the expression pos('z', w), we could check to see if this function returns 0 or a positive number. Zero would mean there is no 'z' in w.

       If we wanted to handle both the capital and the lowercase Z, we would call the pos function twice. We would try pos('z', w) and pos('Z', w). If both of these return zero, then we know there is no Z, either positive or negative. But if either is not zero, then we have some kind of Z.

    b.
```
count := 0;
for i := 1 to length(w) do
   if s[i] = 'z' then
      count := count + 1;
```

    c. We are in luck again. As in part (a) above, we can use the pos function.
       zLocation := pos('z', w);

       However, if we were interested in knowing the location of the first Z whether capital or lowercase, we would have to call pos( ) with both kinds of Z, and compare our answers. For clarity, I will mix in some pseudocode to explain the solution.

```
zCapitalLocation := pos('Z', w);
zLowercaseLocation = pos('z', w);
if zCapitalLocation = 0 and zLowercaseLocation = 0 then
   set the variable zLocation to 0 as well, because neither kind of Z exists.
else if only the zCapitalLocation equals 0,
   set zLocation equal to zLowercaseLocation
else if only the zLowercaseLocation equals 0,
   similarly, set zLocation equal to zCapitalLocation
else – at this point both kinds of Z appear.  We want the earlier one.
   zLocation = the smaller of zCapitalLocation and zLowercaseLocation
```

19. Since we know how many iterations we want, a FOR loop makes the most sense.

```
for i := 1 to 26 do
  writeln(i);
```

```
for i := 26 downto 1 do
  writeln(i);
```

20. Various if-conditions:
    a. (100 <= number) and (number <= 999)
    b. (18 <= number) and (number <= 65)

c. not (18 <= number and number <= 65)
d. number > 0
e. (number mod 5 = 0) and (number mod 4 > 0)
f. number mod 2 = 1
g. (number = 7) or (number = 11) or (number = 13)
h. not ((number = 7) or (number = 11) or (number = 13))

Please note that in parts c and h, the parentheses are used to clarify to us that we are to perform the comparisons first before doing the negation. This is not a function call!

21. Let's say that the variable cents contains a number from 1 to 99.

- NumQuarters := cents div 25, because we want to round down
- cents := cents mod 25
- NumDimes := cents div 10
- cents := cents mod 10
- NumNickels := cents div 5
- NumPennies := cents mod 5

22. This is a comment. It's used by the human reader to understand the purpose of the program, and how the author came up with the algorithm. It's also used during the program to explain points about the program that are not immediately obvious.

23. The output is

2
3
4
5
6
7

24. The expressions evaluate to:
    a. 2.8
    b. 4

    Note that if we had 14 div 5, this would have been 2 rather than 2.8.

25. This loop will have 10 iterations. We have our first value when i equals 1. In this case, 22 – 2i = 20. The last value occurs when i = 10. Then, 22 – 2i = 2. So, this loop will print the even numbers from 20 down to 2, inclusive, one number per line.

26. Run-time error

27. Technically, an isosceles triangle is one that has at least 2 equal sides. In other words, an equilateral triangle is a special kind of isosceles triangle. This means we don't have to worry about excluding this case. It's sufficient to test every pair of sides to see if they are equal.

    First, ask the user for a, b, c, being the sides of a triangle.
    Let's assume that these numbers form a valid triangle.
    If a = b or b = c or a = c,
      Tell the user that the triangle is isosceles
    Else
      Tell the user it's not isosceles.

28. We need to repeat the variable num when checking the mod by 11. In other words, we need to use both the expressions num mod 7 and num mod 11. There is no shortcut to combining them. Also, the := should be =. We can rewrite the statement:

    if (num mod 7 = 0) or (num mod 11 = 0) then

29. The order is: e, b, d, a, c

30. if ((year mod 100 = 0) and (year mod 400 = 0)) or ((year mod 100 > 0) and (year mod 4 = 0)) then
      LeapYear := True
    else
      LeapYear := False;

31. The statement that initializes count to 0 needs to occur before the loop, not inside the loop. Otherwise, the final value of count will either be 0 or 1.

32. B

33. B

34. You could say `while` or `for`.

35. The program does not handle the case of overflow in the hour. The input hour is in the range 1-12. We add 2 to this number. As a result, the output hour is in the possible range 3-14. The output hour values 13 and 14 are incorrect. We need to add an if-statement that says: if the Atlanta hour is greater than 12, then decrement it by 12. This if-statement would be placed immediately after the hour := hour + 2 statement.

36. The three kinds of programming errors:
    a. A syntax error means that we made a mistake when typing in our program. This sort of mistake is a misuse of the programming language. As a result, the computer does not understand what we want it to do. The program will not run unless we fix the error.

b. A run-time error happens when we run the program, and during execution an instruction asks the computer to do something that is impossible. As a result, the system immediately halts our program.
c. A logical error means our program ran without causing a run-time error, but nevertheless the output is wrong because the instructions we gave the computer were incorrect.

37. C

38. 1 + 4 + 7 + 10 equals 22.

39. Run-time error

40. Logical error

41. An algorithm is written in English and a program is written in a programming language such as Pascal.

42. The condition is: (1 <= n) and (n <= 12). You could also have said: (n >= 1) and (n <= 12).

43. The condition is: s[length(s)] = s[length(s) – 1].

44. The condition is: n mod 100 = 65.

45. Round up to the next power of 2 to obtain 32 million, approximately. The number 32 million is 32 times 1 million = $2^5 * 2^{20} = 2^{25}$. Thus, we need 25 bits.

46. Go to the binary store with $44.
    44 – 32 = 12
    12 – 8 = 4
    4 – 4 = 0
    We bought items worth 32, 8 and 4 dollars. Let's fill in a table identifying where these powers of 2 are:

1	0	1	1	0	0
32	16	8	4	2	1

    We see that the binary representation should be 101100.

47. Converting 1101110 from binary to decimal, we identify each "1" as a distinct power of 2, and add them up. Let's first put the bits in the table to more easily see which powers of 2 we have.

1	1	0	1	1	1	0
64	32	16	8	4	2	1

Our decimal answer is 64 + 32 + 8 + 4 + 2 = 110 (i.e. one hundred ten).

To convert to octal, we need to group the original bits into threes. Since the total number of bits (7) is not itself a multiple of 3, it is necessary to pad the left side of the number with extra zeros. Our binary number then becomes: 001 101 110. And we convert each group of 3 bits into its own octal digit.
001 becomes 1
101 becomes 5
110 becomes 6.
Thus, 001 101 110 in binary equals 156 in octal.

To convert to hexadecimal, we use groups of 4 bits. First, we must pad zeros on the left end of the binary number to obtain: 0110 1110. We convert each group of 4 bits into its own hexadecimal digit.
0110 becomes 6
1110 becomes e (because it equals fourteen)
Thus, 0110 1110 in binary equals 6e in hexadecimal.

48. The output is:
    0
    3
    6
    9
    12

49. Here is one way to count the number of times the letter 'A' appears in s:

```
count := 0;
for i := 1 to length(s) do
   if s[i] = 'A' then
       count := count + 1;
```

50. Classifying the examples of errors:
    a. Syntax error
    b. Logical error
    c. Syntax error
    d. Logical error
    e. Logical error
    f. Syntax error

51. We need to simplify the given expression:
    7 div 3 – 1 + 17 mod 5 * 3
    Note that 7 div 3 = 2 and 17 mod 5 = 2. Now, we have:
    2 – 1 + 2 * 3
    Next, we see that 2 * 3 = 6. Now, we have:
    2 – 1 + 6
    Which equals 7

52. Accessing individual characters of a string:
    a. s[5] is ‘c’
    b. s[length(s) – 3] is ‘G’

53. Counts the number of characters in the string s that are either lowercase x, y, or z.

54. The code will determine the number of proper divisors of n. In other words, it finds out which numbers are evenly divisible into n, other than n itself.