Introduction to Computer Science-101
Midterm solution

1. A step-by-step solution to a problem is called __an algorithm__. (5%)

2. In a computer, the _input/output_ subsystem accepts data and programs and sends processing results to output device. (5%)

3. Convert the following decimal numbers to hexadecimal without using a calculator, showing your work. (6%)
   a. 1411 $(583)_{16}$
   b. 16.5 $(10.8)_{16}$

4. Convert the following octal numbers to hexadecimal without using a calculator, showing your work. (6%)
   a. $(13.7)_8$ $(B.E)_{16}$
   b. $(1256)_8$ $(2AE)_{16}$

5. Find the minimum number of digits needed in the destination system for each of the following cases: (6%)
   a. 5-bit binary number converted to decimal. $\lceil \log_{10} 2^5 \rceil = \lceil 1.5 \rceil = 2$
   b. Three-digit octal number converted to decimal. $\lceil \log_{10} 8^3 \rceil = \lceil 2.7 \rceil = 3$

6. Change the following decimal numbers to 16-bit two's complement integers. (6%)
   a. 102 0000 0000 0110 0110
   b. 62056 Over flow because 62,056 is not in the range (−32768, +32767).

7. Convert the following numbers in 32-bit IEEE format. (6%)
   a. -12.640625
      $-12.640625 = (-1100.101001)_2 = -2^3 \times 1.100101001$
      S = 1
      $E = 3 + 127 = 130 = (10000010)_2$
      M = 100101001 (plus 14 zero at the right)
      → 1 10000010 10010100100000000000000
   b. 11.40625
      $11.40625 = (1011.01101)_2 = 2^3 \times 1.01101101$
      S = 0
      $E = 3 + 127 = 130 = (10000010)_2$
      M = 01101101 (plus 15 zero at the right)
      → 0 10000010 01101101000000000000000

8. Show the result of the following operations. (6%)
   a. $(99)_{16}$ OR [NOT $(00)_{16}$]
      $= (10011001)_2$ OR [NOT $(00000000)_2$]= $(10011001)_2$ OR $(11111111)_2$ = $(11111111)_2 = (FF)_{16}$
   b. $(99)_{16}$ OR $(33)_{16}$ AND [$(00)_{16}$ OR $(FF)_{16}$]
      = [$(10011001)_2$ OR $(00110011)_2$] AND [$(00000000)_2$ OR $(11111111)_2$] = $(10111011)_2$ AND $(11111111)_2 = (10111011)_2 = (BB)_{16}$

9. We need to unset (force to 0) the three leftmost bits and set (force to 1) the two rightmost bits of a pattern. Show the masks and the operations. (6%)
   Mask1= $(00011111)_2$ Mask2 = $(00000011)_2$
   Operation: [Mask1 AND $(xxxxxxxx)_2$] OR Mask2 = $(000xxx11)_2$

10. We need to set (force to 1) the four rightmost bits of a pattern. Show the masks and the operations. (6%)
   Mask = $(00001111)_2$
   Operation: Mask OR $(xxxxxxxx)_2$ = $(xxxx1111)_2$

11. Show the result of the following floating-point operations using IEEE_127. (6%)
   a. -344.3125 – 123.5625

$-344.3125 - 123.5625 = -(101011000.0101)_2 - (1111011.1001)_2 = 2^8 \times (1.010110000101)_2 - 2^6 \times (1.1110111001)_2$. These two numbers are stored in floating-point format as shown, but we need to remember that each number has a hidden 1 (which is not stored, but assumed). $E_1 = 127 + 8 = 135 = (10000111)_2$ and $E_2 = 127 + 6 = 133 = (10000101)_2$

	S	E	M
A	1	10000111	01011000010100000000000
B	0	10000101	11101110010000000000000

The first two steps in UML diagram is not needed. Since the operation is subtraction, we change the sing of the second number.

	S	E	M
A	1	10000111	01011000010100000000000
B	1	10000101	11101110010000000000000

We denormalize the numbers by adding the hidden 1's to the mantissa and incrementing the exponent. Now both denormalized mantissas are 24 bits and include the hidden 1's. They should store in a location to hold all 24 bits. Each exponent is incremented.

	S	E	M
A	1	10001000	101011000010100000000000
B	1	10000110	111101110010000000000000

We align the mantissas. We increment the second exponent by 7 and shift its mantissa to the right seven times.

	S	E	M
A	1	10001000	101011000010100000000000
B	1	10001000	001111011100100000000000

Now we do sign-and-magnitude addition treating the sign and the mantissa of each number as one integer stored in sign-and-magnitude representation.

	S	E	Denormalized M
R	1	10001000	111010011111000000000000

There is no overflow in mantissa, so we normalized.

	S	E	Denormalized M
R	1	10000111	11010011111000000000000

The mantissa is only 23 bits because there is no overflow, no rounding is needed.

$$E = (10000111)_2 = 135,\ M = 11010011111$$

The result is

$$(1.11010011111)_2 \times 2^{135-127} = (111010011.111)_2 = 467.875$$

b. 34.75 + 23.125

$34.75 + 23.125 = (100010.11)_2 + (10111.001)_2 = 2^5 \times (1.0001011)_2 + 2^4 \times (1.0111001)_2$. These two numbers are stored in floating-point format as shown, but we need to remember that each number has a hidden 1 (which is not stored, but assumed). $E_1 = 127 + 5 = 132 = (10000100)_2$ and $E_2 = 127 + 4 = 131 = (10000011)_2$. The first few steps in UML diagram is not needed. We move to denormalization. We denormalize the numbers by adding the hidden 1's to the mantissa and incrementing the exponent.

	S	E	M
A	0	10000100	00010110000000000000000
B	0	10000011	01110010000000000000000

Now both denormalized mantissas are 24 bits and include the hidden 1's. They should store in a location to hold all 24 bits. Each exponent is incremented.

	S	E	Denormalized M
A	0	10000101	100010110000000000000000
B	0	10000100	101110010000000000000000

We align the mantissas. We increment the second exponent by 1 and shift its mantissa to the right once.

	S	E	Denormalized M
A	0	10000101	100010110000000000000000
B	0	10000101	010111001000000000000000

Now we do sign-and-magnitude addition treating the sign and the mantissa of each number as one integer stored in sign-and-magnitude representation.

	S	E	Denormalized M
R	0	10000101	111001111000000000000000

There is no overflow in mantissa, so we normalized.

	S	E	M
R	0	10000100	11001111000000000000000

The mantissa is only 23 bits because there is no overflow, no rounding is needed.

$$E = (10000100)_2 = 132,\ M = 11001111$$

In other words, the result is

$$(1.11001111)_2 \times 2^{132-127} = (111001.111)_2 = 57.875$$

12. Show the result of the following operations assuming that the numbers are stored in 16-bit two's complement representation. Show the result in hexadecimal notation. (6%)

a. $(E12A)_{16} + (9E27)_{16}$

																	Carry	Hexadecimal
1											1		1		1	1		
	1	1	1	0	0	0	0	1	0	0	1	0	1	0	1	0		E12A
+	1	0	0	1	1	1	1	0	0	0	1	0	0	1	1	1		9E27
	0	1	1	1	1	1	1	1	0	1	0	1	0	0	0	1		17F51

Note that the result is not valid because of overflow.

b. $(712A)_{16} + (9E00)_{16}$ = $(0F2A)_{16}$

	1 1 1 1				Carry	Hexadecimal
	0 1 1 1	0 0 0 1	0 0 1 0	1 0 1 0		712A
+	1 0 0 1	1 1 1 0	0 0 0 0	0 0 0 0		9E00
	0 0 0 0	1 1 1 1	0 0 1 0	1 0 1 0		10F2A

13. How many bits are needed in a 50-pixel image using True-Color encoding? (6%)
    50*24=1200

14. An imaginary computer has sixteen data registers (R0 to R15), 2048 words in memory, and 32 different instructions (add, subtract, and so on). What is the minimum size of an instruction in bits if a typical instruction uses the following format: Instruction M R2 . (6%)
    4+11+5=20

15. A computer has 128 MB of memory. Each word is 8 bytes. How many bits are needed to address each single word in memory? (6%)
    We have 128 MB / (8 bytes per word) = 16 Mega words = $16 \times 2^{20} = 2^4 \times 2^{20} = 2^{24}$ words. Therefore, we need 24 bits to access memory words.

16. What is the main function of the data-link layer in the TCP/IP protocol suite? What type of addresses is used in this layer? (6%)
    The data link layer delivers a frame from a node to another. Data link layer addresses are often called physical addresses or medium access control (MAC) addresses.

17. What is the main function of the network layer in the TCP/IP protocol suite? What type of addresses is used in this layer? (6%)
    The network layer is responsible for the source-to-destination (computer-to-computer or host-to-host) delivery of a packet, possibly across multiple networks (links). The address used at this level is the logical or IP address.