CSE 101: Introduction to Computational and Algorithmic Thinking

Exam 3 Study Questions

Data Representation and Compression

1. Perform the following base conversions, where the subscript indicates the base you are converting *from*. You must show work to have a chance at partial credit. Circle your final answer to each question.

   a. 101011_2 to base 10

   a. 43_{10}

   b. 67_{10} to base 2

   b. 1000011_2

   c. 1010011011011001_2 to base 16

   c. $A6D9_{16}$

   d. $35A7_{16}$ to base 2

   d. 0011010110100111_2

   e. 1011_2 to base 10

   e. 11_{10}

   f. 110101_2 to base 10

   f. 53_{10}

   g. 10010101_2 to base 10

   g. 149_{10}

   h. 72_{10} to base 2

   h. 1001000_2

   i. 55_{10} to base 2

   i. 110111_2

   j. 103_{10} to base 2

   j. 1100111_2

   k. 0010110101011000_2 to base 16

   k. $2D58_{16}$

   l. 0111010111011110_2 to base 16

   l. $75DE_{16}$

m. $2BA_{16}$ to base 2

m. 1010111010_2

n. $F416_{16}$ to base 2

n. 1111010000010110_2

o. 23.3_{10} to fixed-point binary

o. 10111.01001_2

2. Convert 19.375_{10} to fixed-point binary with 5 digits to the right of the binary point.

   $19_{10} = 10011_2$

   $0.375 \times 2 = 0.75 \rightarrow 0$

   $0.75 \times 2 = 1.5 \rightarrow 1$

   $0.5 \times 2 = 1.0 \rightarrow 1$

   Answer: 10011.01100_2

3. Convert 18.7_{10} to fixed-point binary with 5 digits to the right of the binary point.

   $18_{10} = 10010_2$

   $0.7 \times 2 = 1.4 \rightarrow 1$

   $0.4 \times 2 = 0.8 \rightarrow 0$

   $0.8 \times 2 = 1.6 \rightarrow 1$

   $0.6 \times 2 = 1.2 \rightarrow 1$

   $0.2 \times 2 = 0.4 \rightarrow 0$

   Answer: 10010.10110_2

4. Convert the fixed point value -1011.0011 into IEEE 754 32-bit notation.

   Sign bit = 1

   Exponent = $3 + 127 = 130_{10} = 10000010_2 \rightarrow$ `10000010`

   Mantissa = `01100110000000000000000`

   Final answer: `1 10000010 01100110000000000000000`

5. Convert the fixed point value 0.0101101 into IEEE 754 32-bit notation.

   Sign bit = 0

   Exponent = $(-2) + 127 = 125_{10} = 1111101_2 \rightarrow$ `01111101`

   Mantissa = `01101000000000000000000`

   Final answer: `0 01111101 01101000000000000000000`

6. Convert the IEEE 754 32-bit representation `00111110011011000000000000000000` into fixed-point binary.

   Divide into parts: `0 01111100 11011000000000000000000`

   Sign bit is 0 $\rightarrow$ positive number.

   Exponent = $01111100_2 = 64 + 32 + 16 + 8 + 4 = 124 \rightarrow 124 - 127 = -3$

   Mantissa = $1.11011_2 \times 2^{-3} = 0.00111011_2$

7. Convert the IEEE 754 32-bit representation `01000001111001000000000000000000` into decimal.

   Divide into parts: `0 10000011 11001000000000000000000`

   Sign bit is 0 $\rightarrow$ positive number.

   Exponent = $10000011_2 = 128 + 2 + 1 = 131 \rightarrow 131 - 127 = 4$

   Mantissa = $1.11001_2 \times 2^2 = 11100.1_2 = 16 + 8 + 4 + 0.5 = 28.5$

8. What would be the parity bit for the bitstring `1101011`? Why?

   The parity bit would be `1` so that the transmitted bitstring would have an even number of 1s in it.

9. a. Give the Huffman codes for the letters below, given the frequencies provided in the table:

Letter	Frequency
A	0.35
B	0.10
C	0.18
D	0.22
E	0.15

```
0.10    0.15    0.18    0.22    0.35
B-------E-------C-------D-------A

0.18    0.22    0.25    0.35
C-------D-------x-------A
               / \
              B   E

0.25   0.35     0.40
  x----A--------x
 / \           / \
B   E         C   D

0.40        0.60
  x----------x
 / \        / \
C   D      x   A
          / \
             B   E
```

```
          x
       0/   \1
      x       x
   0/ \1   0/ \1
   C   D   x   A
         0/ \1
         B   E
```

Letter	Huffman Code
A	11
B	100
C	00
D	01
E	101

b. What would be the encoding for the string “BED”?
10010101

c. What word would be encoded as 011011101?
DEAD

10. Give the Huffman codes for the letters below, given the frequencies provided in the table:

Letter	Frequency
A	0.20
B	0.32
C	0.25
D	0.15
E	0.08

```
0.08    0.15    0.20    0.25    0.32
E-------D-------A-------C-------B

0.20    0.23    0.25    0.32
A-------x-------C-------B
       / \
      E   D

0.25    0.32     0.43
C-------B--------x
                / \
               A   x
                  / \
                 E   D

0.43         0.57
  x----------x
```

```
 / \        / \
A   x      C   B
   / \
  E   D

      x
  0/      \1
  x        x
0/ \1    0/ \1
A   x    C   B
  0/ \1
  E   D
```

Letter	Huffman Code
A	00
B	11
C	10
D	011
E	010

Bitwise Operators

11. Give the value of each expression below, which involve bitwise operators and binary numbers.

    a. `0011010010101101 AND`
    `0011011010110111`

    a. `0011010010100101`

    b. `0011010010101101 OR`
    `0011011010110111`

    b. `0011011010111111`

    c. `0011010010101101 XOR`
    `0011011010110111`

    c. `0000001000011010`

    d. `0011010010101101 >> 2`

    d. `0000110100101011`

    e. `1100001111010101 << 4`

    e. `0011110101010000`

12. Give both the operation (`OR`, `AND`, or `XOR`) and the 16-bit mask that would turn off bits 2, 3 and 7, leaving other bits unchanged.

AND 1111111101110011

13. Give both the operation (OR, AND, or XOR) and the 16-bit mask that would turn on bits 2, 3 and 7, leaving other bits unchanged.

    OR 0000000010001100

14. Give both the operation (OR, AND, or XOR) and the 16-bit mask that would toggle bits 2, 3 and 7, leaving other bits unchanged.

    XOR 0000000010001100

Regular Expressions

Before attempting these problems, complete the tutorials on the website www.regexone.com. Running through the tutorials won't take you that long, and you will have a good understanding how how to write them by the time you have finished.

For regex practice exercises you should visit www.regex101.com and try your solution there. For many situations there are at least several regular expressions that can be used to solve a given problem. Experiment and try to write your answer in multiple ways.

15. For each of the following strings, write a regular expression that would match *only* those strings that match the described patterns.

    a. A SBU student ID number consisting of 9 digits and starting with the digits 11 and ending with the digits 22.

       `11\d{5}22`

    b. An integer in the range 100 through 599, inclusive.

       `[1-5]\d\d`

    c. A email address which consists of 6 uppercase letters, lowercase letters and digits (in any order), followed by an at sign (@), followed by 5 lowercase letters, followed by .edu.

       `[A-Za-z0-9]{6}@[a-z]{5}\.edu`

    d. A birth date in the format mm-dd-yyyy, where mm is 01 through 12 (inclusive), dd is in the range 01 through 31, and yyyy is in the range 1900 through 1999, inclusive. Include leading zeroes to make mm and dd two digits each. Don't worry about invalid dates, so a string like 02-30-1977 would be acceptable.

       `(0[1-9]|1[0-2])-((0[1-9])|([1-2]\d)|(3[0-1]))-19\d\d`

    e. Strings consisting of one or more decimal digits.

       `\d+`

    f. Strings consisting of any mixture of at least five uppercase and/or lowercase letters.

       `[a-zA-Z]{5,}    or    [a-zA-Z]{5}[a-zA-Z]*`

    g. Product codes that have the following format:

       1. The first two symbols are AT or HH.

2. Next comes the number 8.
3. Then comes any combination of exactly two lowercase letters.
4. The product code ends with MG or ML.

`(AT|HH)8[a-z]{2}M[GL]`

h. A regular expression consisting of no more than 30 characters that will match 24-hour times of the following format, and only this format: HH:MM, where HH is a two-digit number in the range 00-23 (inclusive) and MM is a two-digit number in the range 00-59 (inclusive).

`(([01]\d)|(2[0-3])):[0-5]\d`

i. A regular expression consisting of no more than 20 characters that will match U.S. zip codes of the form NNNNN or NNNNN-NNNN, where N is a digit in the range 0 through 9 (inclusive).

`\d{5}(-\d{4})?`

j. On the Internet, IP addresses are represented in dotted-quad format, which means that they contain exactly four integers, each of which ranges from 0–255, inclusive, separated by periods (for example, 247.125.106.11 is a valid IP address). Write a regular expression that will match only valid IP addresses written in dotted-quad format.

A single term (an integer between 0 and 255) can be represented by the regular expression

`((1\d{2})|(2(([0-4]\d)|(5[0-5])))|([1-9]\d)|\d)`

so we repeat this pattern (followed by a literal period) three times, plus once more without the trailing period to get:

```
(((1\d{2})|(2(([0-4]\d)|(5[0-5])))|([1-9]\d)|\d)\.){3}
 ((1\d{2})|(2(([0-4]\d)|(5[0-5])))|([1-9]\d)|\d)
```

k. In Web design, RGB (red-green-blue) colors are often represented using hexadecimal: a hash sign (#) is followed by three two-digit hexadecimal values (describing the red, green, and blue values respectively). For example, #FF0000 represents pure red. Write a regular expression that will ONLY match values in this format.

`#[A-F0-9]{6}`

l. State-issued license plates for cars generally follow a simple 3-4 pattern: three uppercase letters, followed by a single dash, followed by four digits. Write a regular expression that will match valid license plates that match this pattern, and only those license plates that match this pattern.

`[A-Z]{3}-\d{4}`

Cryptography

16. The following message was encrypted using a **Caesar cipher** in which the original alphabet was shifted to the right by 5 letters to create the cipher alphabet. Decrypt the message. Message: LTTI QZHP TS KNSFQX

GOOD LUCK ON FINALS

17. Encrypt the word TRUTH using the **Caesar cipher** with a key of 6.

ZXAZN

18. Encrypt the word SEAWOLF using a **multiplicative cipher** with $k = 5$.

MUAGSDZ

19. Encrypt the word FALSE using the **affine cipher** with $a = 3, b = 2$.

    RCJEO

20. Encrypt the word TEAMWORK using an **affine cipher** with $a = 7$, $b = 3$.

    GFDJBXSV

21. Encrypt the words LONGISLAND using a **rail fence cipher** with a grid of 4 rows.

```
L     L
 O   S A
  N I   N
   G     D

LLOSANINGD
```

22. Encrypt the phrase GOSEAWOLVES using a **rail fence cipher** with a grid of 4 rows.

    Answer: GOOWLSAVSEE

    Here's the grid. Read the letters across each row.

G						O				
	O				W		L			
		S		A				V		S
			E						E	

23. Encrypt the word COMPUTER using a **rail fence cipher** with a grid of 3 rows.

    Answer: CUOPTRME

C				U			
	O		P		T		R
		M				E	

24. The ciphertext WBARRFOETFFASLSKAEF was generated with a **rail fence cipher** by encrypting a plain-text message on a grid with 6 rows. Decrypt it and write the original message.

    To fit the letters into a grid, first create an empty grid:

Then put a placeholder where each letter will ultimately go:

x										x								
	x								x		x							
		x						x				x						x
			x				x						x				x	
				x		x								x		x		
					x										x			

Now read off the letters from the ciphertext one character at a time. Pass through each row, top-to-bottom, left-to-right, replacing each placeholder with the next letter from the ciphertext. Below is the final grid. Read the letters diagonally in zigzag order:

W										B								
	A								R		R							
		F						O				E						T
			F				F						A				S	
				L		S								K		A		
					E										F			

Yummy!

25. Encrypt the message IMASEAWOLF using a **Vigenère cipher** where the keyword is PROGRAM.

On the exam you will be given the following grid:

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Down the left-hand side of the table mark the rows of the table you will need, in the ordered needed.

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
6	**A**	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
4	**G**	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
7	**M**	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
3,10	**O**	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1,8	**P**	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
2,5,9	**R**	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Now write out the letters of the plaintext message in one row and underneath write the letters of the keyword, wrapping around the keyword letters as needed:

I	M	A	S	E	A	W	O	L	F
P	R	O	G	R	A	M	P	R	O

Go to the row marked P. Find the column for I. In that grid cell we find an X. This is the first letter of the ciphertext.

Now go to the row marked R. Find the column for M. In that grid cell we find a D. This is the second letter of the ciphter text.

Continue in this fashion until all letters are encrypted.

Final answer: XDOYVAIDCT

26. Use the table below to encrypt the message RECURSION using a **Vigenère cipher** where the keyword is WHILE.

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
W X Y Z A B C D E F G H I J K L M N O P Q R S T U V
H I J K L M N O P Q R S T U V W X Y Z A B C D E F G
I J K L M N O P Q R S T U V W X Y Z A B C D E F G H
L M N O P Q R S T U V W X Y Z A B C D E F G H I J K
E F G H I J K L M N O P Q R S T U V W X Y Z A B C D
```

NLKFVOPWY

27. The following message was encrypted using a **Vigenère cipher** where the keyword was ILOVEPYTHON. Decrypt the message: AEIYCWYKK

STUDYHARD

Limits of Computation

No answers are provided for these questions, but you should be able to answer them by reviewing the lecture notes and/or reading the relevant chapters of the textbook.

28. What is the *primary* significance of the **Halting Problem**?

29. How is the **Turing Test** organized and administered?

30. If you were the judge/interrogator in a **Turing Test**, what kinds of questions would you ask and why?

31. What is the objective in the **Traveling Salesman Problem** (TSP)?

32. What is an **evolutionary algorithm**? Why did we attempt to solve the **Traveling Salesman Problem** using such an algorithm?

Miscellaneous Review Questions

No answers are provided for these questions, but you should be able to answer them by reviewing the lecture notes and/or reading the relevant chapters of the textbook.

33. What are **ASCII** and **Unicode**? Which is better than the other and why?

34. Explain the major differences between **assembly language** and **high-level programming languages** like Python.

35. Briefly describe the **stored-program concept**. Why is it important?

36. Briefly explain what occurs during a computer's **fetch-decode-execute cycle**.

37. Briefly describe the major components of the **von Neumann architecture** and how they cooperate to form a functioning computer.

38. Briefly describe the major components of a **central processing unit**.

Programming Questions: Short Answer

39. Give the output for each of the following code fragments. Note that there are no syntax errors in any of this code.

a.
```
a = 5 + 23 // 4
print(a)
```

a. ______10______

b.
```
b = 29 % 7 + 12 // 2
print(b)
```

b. ______7______

c.
```
c = 10
if c < 5:
    c += 4
elif c > 8:
    c += 5
else:
    c += 3
print(c)
```

c. ______15______

d.
```
d = 'StonyBrookU'
for i in range(1, 8, 2):
    print(d[i])
```

d. ____ t n B o ____

e.
```
e = [1, 3, 4, 2, 6, 5, 9, 7, 8]
i = 8
while i > 1:
    print(e[i-1])
    i -= 3
```

e. ____ 7 6 3 ____

f.
```
f = [i//3 for i in range (2,12,2)]
print(f)
```

f. ____ [0, 1, 2, 2, 3] ____

g.
```
g = ['a', 't', 'd', 'f', 'q', 'p', 'y', 'r']
print(g[2:-3:2])
```

g. ____ ['d', 'q'] ____

h.
```
def func(n):
    return 2*n - 1

f = 3
for i in range(3):
    f = func(f)
    print (f)
```

h. ____ 5 9 17 ____

i.
```
h = 'Science'
count = 0
for i in range(1,4):
    for j in range(3,7):
        if h[i] == h[j]:
            count += 1
print(count)
```

i. ____ 3 ____

j.
```
letters = [chr(c) for c in range(ord('A'), ord('F'))]
print(letters)
```

j. ____ ['A', 'B', 'C', 'D', 'E'] ____

Programming Problems

You are strongly encouraged to complete the review programming problems from earlier in the course if you haven't done so yet!

40. In web design, hexadecimal values are often used to denote colors. For example, the notation `#E3CA92` would represent the triple (`E3`, `CA`, `38`), which denotes the (red, green, blue) *components* of the color. The `#` symbol merely indicates that hexadecimal is being used to denote a color. The hexadecimal triple (`E3`, `CA`, `38`) written in decimal would be expressed (`227`, `202`, `56`).

    Write a function `h2d_color(hex)` that takes a string such as `"#E3CA38"` and returns a tuple of decimal equivalents for the three color components (the tuple would be (`227`, `202`, `56`) in this example).

```
def h2d_color_helper(hex_digit):
    if hex_digit in 'ABCDEF':
        return ord(hex_digit) - ord('A') + 10
    else:
        return int(hex_digit)

def h2d_color(hex):
    h1 = hex[1:3]
    h2 = hex[3:5]
    h3 = hex[5:7]
    d1 = h2d_color_helper(h1[0])*16 + h2d_color_helper(h1[1])
    d2 = h2d_color_helper(h2[0])*16 + h2d_color_helper(h2[1])
    d3 = h2d_color_helper(h3[0])*16 + h2d_color_helper(h3[1])
    return d1, d2, d3
```

41. Write a function called `encrypt(text)` that takes a string argument consisting of lowercase letters only and returns an encrypted version of that string. The function encrypts a word by replacing each letter with the character two places ahead of it in the alphabet. The last two letters of the alphabet wrap around to the front (i.e., "y" → "a", "z" → "b"). Assume that all input consists solely of lowercase letters, and that only one word is input. (Hint: Python has a built-in function `ord(c)` that returns the ASCII character code of its input. So, `ord('a')` returns 97, `ord('b')` returns 98, and so forth. Python also has a built-in function `chr(n)` that is the inverse of `ord(c)`. Thus, `chr(97)` returns 'a', and so forth.)

```
def encrypt(text):
    result = ''
    for c in text:
        n = ord(c) - ord('a')
        n = (n + 2) % 26 + ord('a')
        new_c = chr(n)
        result += new_c
    return result
```

42. Write a Python function `count_letters` that takes a list of strings as the argument `license_plates` and returns a dictionary that maps each letter to its frequency in the argument strings. Each string contains some combination of 7 or 8 uppercase letters, spaces and digits. Digits and spaces are ignored while building the dictionary. For example, for the following list:

```
license_plates = ['AHE 3123', 'HE2 91SS', 'HH2299 2', 'SBU2017']
```

    the function would return this dictionary:

```
{'A': 1, 'H': 4, 'E': 2, 'S': 3, 'B': 1, 'U': 1}
```

Hint: the method `isalpha()` returns `True` if a string contains only letters.

```
def count_letters(license_plates):
    result = {}
    for chars in license_plates:
        for c in chars:
            if c.isalpha():
                result.setdefault(c, 0)
                result[c] += 1
    return result
```

Programming Questions: Classes and Objects

43. You are given the following class definition that represents a book:

```
class Book:
    def __init__(self, title, author, price):
        self._title = title         # string
        self._author = author       # string
        self._price = price         # integer
```

a. Write a single line of code that would construct (create) a new `Book` object that represents author George Orwell's book "Animal Farm", which sells for $10. (Don't include the dollar sign.) Store the object in a variable named `farm`.

```
farm = Book("Animal Farm", "George Orwell", 10)
```

b. Write a complete function `avg_cost(books)` that takes a list of `Book` objects and returns the average price of the books in the list.

```
def avg_cost(books):
    total = 0
    for b in books:
        total += b._price
    return total / len(books)
```

c. Write a complete function `cheapest_book(books)` that takes a list of `Book` objects and returns the title of the cheapest book in the list.

```
def cheapest_book(books):
    title = 0
    min_price = books[0]._price
    for b in books:
        if b._price < min_price:
            title = b._title
            min_price = b._price
    return title
```

44. You are given the following class definition that represents a house:

```
class House:
    def __init__(self, price, year):
        self._price = price
        self._year = year
```

Write a complete function `house_sort(houses)` that takes a list of `House` objects and transforms the list by sorting the houses using the "gnome sort" algorithm, putting the houses into *descending* order by price. So, the most expensive house will be first in the list, the second-most expensive house will be second, etc., with the cheapest house last. Recall that the gnome sort algorithm for putting a list of numbers into *ascending* order has the following pseudocode:

```
procedure gnome_sort(nums):
    pos = 0
    while pos < length(nums):
        if (pos == 0 or nums[pos] >= nums[pos-1]) then
            increment pos by 1
        otherwise
            swap nums[pos] and nums[pos-1]
            decrement pos by 1
```

You may add methods to the `House` class if you like, but doing so is not necessary to write the `house_sort()` function. Your `house_sort()` function MUST implement the "gnome sort" algorithm.

```
def house_sort(houses):
    pos = 0
    while pos < len(houses):
        if pos == 0 or houses[pos]._price <= houses[pos-1]._price:
            pos += 1
        else:
            houses[pos], houses[pos-1] = houses[pos-1], houses[pos]
            pos -= 1
```

45. Consider the code below. The `Waiter` class represents a restaurant waiter, who has a name and a list of orders (`Order` objects) he has taken from customers. An `Order` object is defined by the item a customer has ordered and any special instructions (`instr`) the customer gave.

```
class Waiter:
    def __init__(self, name):
        self.name = name
        self.orders = []

    def take_order(self, item, instr):
        # Complete this method for part (a)

class Order:
    def __init__(self, item, instr):
        self.item = item
        self.instr = instr

def count_orders_of(waiters, item):
    # Complete this function for part (b)
```

a. Complete the `take_order()` method, which constructs an `Order` object from the arguments given and appends the new object to the end of the `orders` field of a `Waiter` object.

```
self.orders.append(Order(item, instr))
```

b. Complete the `count_orders_of()` function, which takes a list of `Waiter` objects and a string representing the name of an item that some customers might have purchased. Using the arguments, the function counts and returns the number of times that customers ordered that `item` from the `waiters`.

```
    total = 0
    for w in waiters:
        for o in w.orders:
            if o.item == item:
                total += 1
    return total
```