Midterm Exam 2B Answer key

15110 Principles of Computing Fall 2015

April 6, 2015

Name: ______________________________

Andrew ID: ____________ Lab section: ____________

Instructions

- Answer each question neatly in the space provided.
- There are 6 questions totaling 31 subproblems on 12 pages. Not all problems are the same size or difficulty.
- Please read each question carefully. You have 50 minutes for this exam. No electronic devices allowed. Good luck!

	Max	Score
1	14	
2	22	
3	19	
4	25	
5	14	
6	6	
Total:	100	

Questions

1. (6 points) This question deals with recursion and recursive algorithms. Here is a function definition in Python that uses recursion.

```
def bad(n):
    if n <= 1:
        return True
    elif n % 2 == 0:
        return bad(n/2)
    else:
        return bad(n/1)
```

(a) (1 point) What is the value returned by `bad(8)`?

Solution: `True`.

(b) (3 points) Think about what happens if we evaluate `bad(3)` . Either give the value that would be returned and displayed by Python, or explain why no value would be returned and displayed.

Solution: No value would be returned because the sequence of function calls would be `bad(3), bad(3), ...`, because we start with an odd argument, which is divided by one, yielding an endless series of identical function calls.

(c) (4 points) Consider the following Python function g, defined for an integer n recursively as follows:

```
def g(n):
    print(n, end=" ")
    if n < 3:
        return n
    else:
        return g(n // 2) - g(n // 3)
```

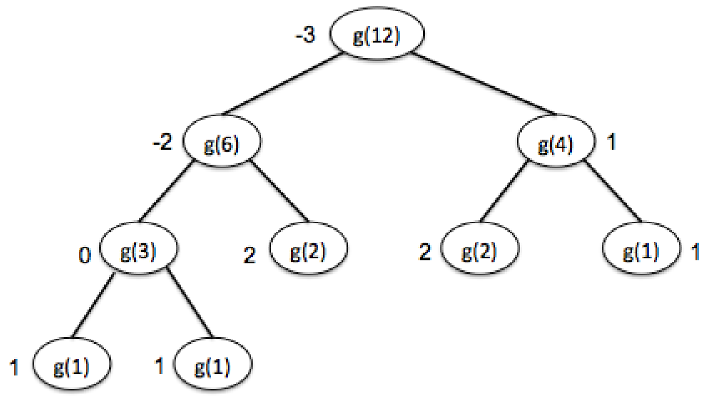
Compute $g(12)$ by drawing a recursion tree showing all of the recursive calls and then use your tree to compute the final answer.

$g(12) =$

Solution: -3

Recursion tree:

Solution:



2. This problem focuses on the representation of data in a computer. The following table may be helpful:

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1

(a) (2 points) What is the largest number that can be represented as a *signed two's complement* integer using 5 bits? Give your answer in decimal notation, not binary.

Solution: 15

(b) (2 points) What is the smallest (most negative) number that can be represented as a *signed two's complement* integer **using 4 bits?** Give your answer in decimal notation, not binary.

Solution: -8

(c) (3 points) Compute the decimal value of the byte 10011001 if it is interpreted as a signed two's complement integer. Show your work.

Solution:
Explanation: the sign bit is set, so it is negative. One's complement: 01100110, plus 1, gives 01100111. So the magnitude is $64 + 32 + 4 + 3 + 1 = 103$, or $6 \times 16 + 7 = 103$.

(d) (2 points) When using odd parity, what is the correct parity bit for the 7-bit character 101 1011 ?

Solution: 0

(e) (2 points) Suppose that the rightmost bit is corrupted during transmission of the eight bits from part (d) and is flipped from 1 to 0.
Which of the following is true? Circle the appropriate letter:

Solution: B

A. The receiver cannot detect the error.
B. The receiver can detect the error but cannot determine which bit is wrong.
C. The receiver can detect the error and can correct the bit that is wrong.

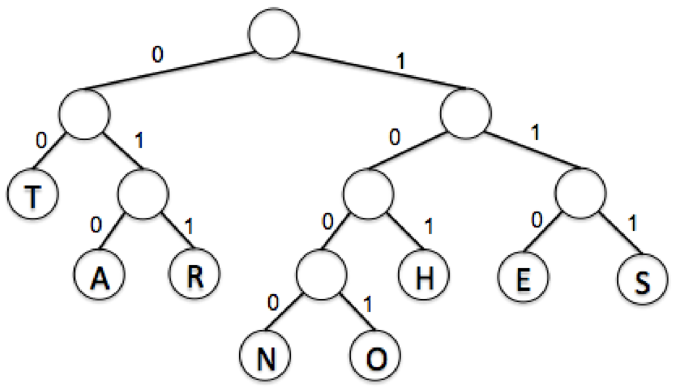
(f) (2 points) Suppose we're sending ASCII codes with a parity bit, using odd parity on an unreliable communication channel. With a reliability of 75%, our data source has a certain measurable entropy (information content). If the reliability of the communication channel goes up to 85%, will the entropy increase or decrease?

Solution: decrease

(g) (2 points) Suppose you are given a frequency table for typical text in Russian using the 33 letters of the Cyrillic alphabet, and you calculate an information entropy value of 5.4 based on the table. If you use the same table to create a Huffman tree and use that tree to compress a typical Russian text of 100,000 letters, **approximately** how many bits would you expect to need to represent the compressed text? You should assume that the frequencies are accurate for the text that is compressed. Show your work.

Solution: Multiply the entropy by the number of characters, so 540,000 bits.

Consider the following Huffman tree:



(h) (4 points) What word is represented by the following binary string, based on the Huffman tree (we have inserted spaces purely for readability; they do **not** indicate boundaries between letters): 111 110 100 011 110 010 11

Solution: SENSOR

(i) (3 points) Show the encoding of the string "TAR" using the same Huffman tree. Leave spaces between the codes to indicate the boundaries between letters.

Solution: 00 010 011

3. This question is concerned with generating and using pseudorandom numbers.

   Recall that the Python function `randint(n, m)` returns a random integer between n and m, inclusive. Using the `randint` function, show how to compute the following using **one** Python expression:

   (a) (3 points) A random integer between 1 and 22 that is a multiple of 4. Write one expression using `randint`.

   Solution: `randint(1, 5) * 4` or something equivalent

   (b) (3 points) A randomly chosen number from the list `vals` below. You are not allowed to use variables other than `vals` or any function other than `randint` or `len`.

   `vals = [17.0, 7.7, 3.1, 5.0, 4.8, 2.6]`

   Solution: `vals[randint(0, len(vals) - 1)]` or `names[randint(0, 5)]`

   (c) (4 points) Write a function called `coin()` that simulates a coin that comes up heads 60% of the time and tails 40% of the time. The function should return a string, either "H" or "T".

   Solution:

```
def coin():
    random_v = randint(0,99) % alternative: randint(1,100)
%                              or randint (0,9), etc.
    if random_v < 60:        % and random_v <= 60
%                              or random_v < 6, etc.
        return "H"
    else:
        return "T"
```

   (d) (2 points) Describe in one or two sentences what method you could use to determine whether your code for part c behaves the way it should.

   Solution: Use a Monte Carlo method: call coin() a large number of times, count the number of heads, and calculate the proportion of all flips that are heads.

   Recall the linear congruential generator (LCG) formula:

$$x_{(i+1)} = (a \times x_i + c) \mod m \qquad \text{(for positive integers } a, c, \text{ and } m)$$

   As described in class, there are certain conditions that should be obeyed for the LCG to have its maximum period:

   1. c and m must be relatively prime;
   2. $a - 1$ must be divisible by every prime factor of m; and
   3. if m is divisible by 4, then $a - 1$ must be divisible by 4.

(e) (4 points) Consider *only* values of a that are **less than** m **(We intended to consider only values larger than 1, but forgot to say that. Alternate acceptable answers below are in boldface.** Assume that $c = 4$ and $m = 45$ (note that the prime factors of 45 are 3 and 5). List the possible values of a that yield the maximum period.

Solution: For $a - 1$, Rule 2 gives us the multiples of 15 less than 45: **0,** 15, 30. Rule 3 does not apply. So a can be 16 or 31 **or 1**.

(f) (3 points) With $x_0 = 0$, $a = 7$, $c = 3$, and $m = 18$, what are the values of x_1, x_2, and x_3? That is, starting with a seed of 0, what are the next three values calculated by the LCG? (Don't worry about the period of this LCG; it doesn't matter whether it has the maximum possible period.)

Solution: 3, 6, 9

4. This question deals with data organization.

(a) (3 points) An airline company wants to maintain a database of up to 3000 frequent flyer numbers of people who use their company most frequently so that it can be determined very quickly whether or not a given frequent flyer number is in the database before they serve the customer. Suppose that the speed of response is very important but efficient use of memory is not as important. Which of the following data structure would be most appropriate for this task? Circle the appropriate letter and explain why in at most two sentences.

Solution: A. The frequent flyer number can be used as an index into the array, giving a constant lookup time. In the worst-case, the lookup time in a linked list is linear in the number of entries.

A. an array with 3000 entries

B. a linked list

(b) (2 points) Suppose that we wish to store a collection of key and value pairs using a hash table. What is the most important property we want for the hash function so that we have efficient search time?

Solution: It should distribute the keys across the table as evenly (uniformly) as possible. It should also have O(1) time complexity.

(c) (3 points) Why are collisions undesirable in a hash table?

Solution: A collision causes multiple data elements to be stored in a single bucket. The lookup time increases as the number of items in a bucket increases.

(d) (4 points) The inputs `d` and `lst` to function `mystery` below are, respectively, a dictionary and a list.

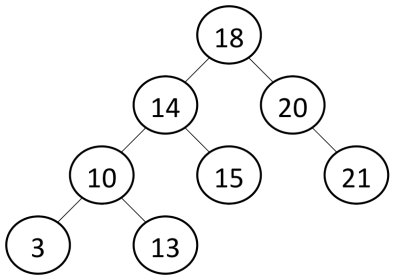
```
def mystery(d,lst):
   s = 0
   for k,v in d.items():
        if k in lst:
            s = s + d[k]
   return s
```

Suppose that we execute the following commands in the given order. What does the function call return?

```
>>> myd = {"x":10, "y": 15, "z":20, w": 5}
>>> mylst = ["x","w"]
>>> mystery(myd,mylst)
```

Solution: 15

(e) (4 points) The binary search tree shown below was constructed by inserting a sequence of items with integer keys into an empty tree.



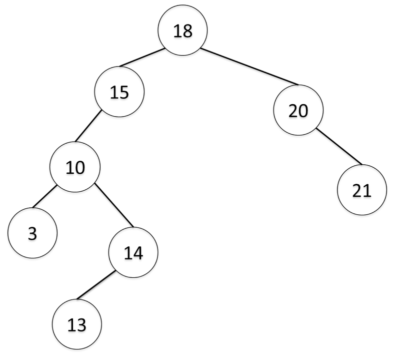
Which of the following input sequences will **not** produce this binary search tree? Circle your answer.

A. 18 20 21 14 10 15 3 13
B. 18 14 20 10 15 21 3 13
C. 18 14 20 10 15 21 13 3
D. 18 15 29 10 14 3 13 21

Solution: D

(f) (4 points) Draw the binary search tree that would result from the insertion order you circled in the previous part.

Solution:



(g) (3 points) In general for a binary tree with 127 nodes, what is the minimum number of levels the tree can have? Assume that a tree consisting of a single node is considered to have one level.

Solution: 7

(h) (2 points) If we store a collection of items in a binary search tree, why is it desirable to construct a tree using as small a number of levels as possible?

Solution: Because the search time is proportional to the number of levels in the tree.

5. This question deals with Boolean logic, gates and computer organization.

(a) (4 points) Write the truth table for the Boolean expression $\neg A \vee \neg B$, where A and B are Boolean variables. Hint: Think about how many rows you need to have in your table.

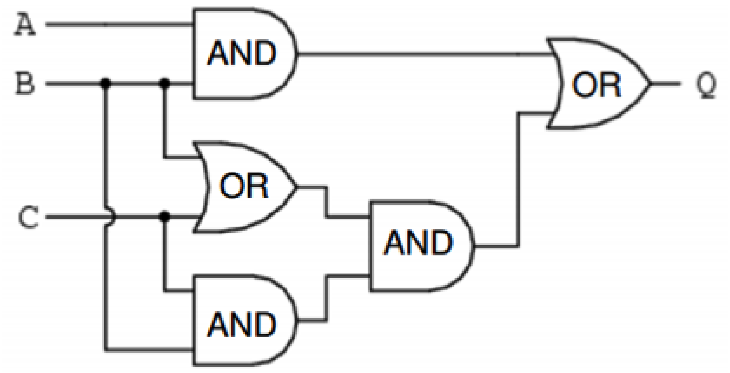
Solution:

A	B	$\neg A$	$\neg B$	$\neg A \vee \neg B$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

(b) (4 points) The truth table you have written above can be implemented using a single gate with the inputs A and B. Which of the following gates is it? AND, OR, NOT, NAND, NOR, XOR. Explain your reasoning.

Solution: NAND. First, transform the expression using De Morgan's Law to $\neg(A \wedge B)$. This is directly implementable using a NAND gate.

(c) (3 points) The circuit below computes the Boolean function Q give input A, B, C and D. What is output on the Q line when A = 1, B = 1, and C = 1?



Solution: 1

(d) (3 points) The following is a description of how a program is executed in a typical computer that conforms to the von Neumann architecture. Fill in the blanks.

```
Load the program counter to point to the first instruction in memory.
Then, repeatedly

1. _______________  the instruction indicated by the program counter
from memory into the instruction register.

2. ________________ the instruction to a control signal and get any
data it needs (possibly from memory).

3. _________________ the instruction with data in ALU and store results
(possibly into memory), incrementing the program counter at the end
```

Solution:
Load the program counter to point to the first instruction in memory. Then, repeatedly

- **Fetch** the instruction indicated by the program countre from memory into the instruction register.
- **Decode** instruction to a control signal and get any data it needs (possibly from memory).
- **Execute** instruction with data in ALU and store results (possibly into memory, incrementing the program counter at the end

6. This question deals with simulation.

   (a) (6 points) Recall the simulation for the spread of a flu virus in a population we did in class. In the simulation code the constants `HEALTHY` and `IMMUNE` are used to represent, respectively, the immune and healthy states of an individual. The population is represented using a two-dimensional list called `matrix` where all elements of `matrix` are of the same length. The following is a Python function to check whether the simulation has reached a state where the virus cannot spread any further because everyone is either immune or healthy. Fill in the blanks. Note that we did not write this function in class. Hint: The function will return True only if the function has examined every cell in the matrix and has not found anyone that is in a state other than immune or healthy.

```
def all_good(matrix):

    for i in range(0,len(matrix)):

        for j in  ______________________:

            if ________________________________________________:

                return False

    return True
```

Solution:

```
def all_good(matrix):
    for i in range(0,len(matrix)):
        for j in  range(0,len(matrix[0])): # Alternatively len(matrix[i])
            if matrix[i][j] != IMMUNE and matrix[i][j] !=HEALTHY:
            # Alternatively, not (matrix[i][j] == IMMUNE or matrix[i][j] == HEALTHY
            # Or something equivalent
                return False
    return True
```