CS115 Fall 2014 Test 2 —SOLUTION GUIDE—

Closed book: no textbook, no electronic devices, one sheet of paper with notes. Hand in your note sheet, with your name on it please.

Read carefully before answering! Write your answers on the test paper.

Question 1 *(15 points)* **Assess:** [encoding]

(a) What is the binary representation of twenty three (i.e., 23_{10})? Write it using exactly 8 bits.

SOLUTION 00010111

Rubric: 2 for 8 bits, 3 for correct answer

(b) Using two's complement with exactly 8 bits, what is the binary representation of negative eighteen? (i.e. -18_{10}).

SOLUTION 11101110

Because: 18 is 00010010 so to get the negation we flip bits to get 11101101 and then add 1 to get 11101110.

Rubric: 5 for correct answer, or else 2 if not correct but show calculation and it is sensible

(c) Using two's complement with 5 bits, what is the largest positive integer that can be represented? What is the smallest (most negative)? Please write your answers in base 10.

SOLUTION

The largest is $2^{(5-1)} - 1$ which is 15. The smallest is $-2^{(5-1)}$ which is -16. Alarmingly, some people wrote things like "$2^{(5-1)} - 1$ which is 31" —which it isn't.

Rubric: 2 max in some form + 2 min in some form + 1 for nothing erroneous

Question 2 *(5 points)* **Assess:** [execution]

What gets printed by this code?

```
>>> score = {}
>>> score['Alanis'] = 76
>>> score['Ani'] = 80
>>> score['Alanis'] = 95
>>> print score.has_key('Neneh')
>>> print score['Alanis']
```

SOLUTION

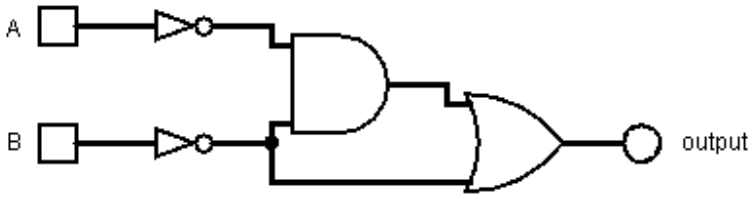
```
False
95
```

NOTE about Python: An alternative to the expression `score.has_key('Neneh')` is the expression `'Neneh' in score`. When used with dictionaries, the "in" operator checks whether an item is in the set of keys in the dictionary.

Rubric: 3 each for the two values, up to 5

Question 3 *(10 points)*

Look at this circuit diagram.



(a) Write out the boolean expression for the circuit in Python. Use variables A and B combined with Python's logical operators.

SOLUTION (not A and not B) or not B

Another acceptable solution is $\overline{A}\,\overline{B} + \overline{B}$, as that's the notation used in the texbook and slides.

Note: during the test it was clarified that the rightmost circle isn't a gate; to write "nor", the little circle must be adjacent to the "or" gate.

Rubric: TODO

(b) What is the output when A=0 and B=1 ?

SOLUTION 0 (or you can say False)

Rubric: 5 each for parts (a) and (b)

Question 4 *(15 points)* **Assess:** [design]

Implement the following function, using recursion on `L`. You are only allowed to access `L` using the expressions `L[0]`, `L==[]`, and `L[1:]`.

```
def dropWhile(f, L):
    '''
    Assume L is a list and f is a function that returns True or False.
    Discard the elements of f that make f true, up to but not including
    the first element that makes f false.'''
```

For example, `dropWhile(odd ,[1,3,2,5,3])` is `[2,5,3]` (assuming `odd` does what its name suggests). Also, the following tests should all print True.

```
def testDropWhile():
    '''Prints True for each successful test.'''
    print dropWhile(lambda x: x>0, [1,7,0,1,7]) == [0,1,7]
    print dropWhile(lambda x: x>0, [-2,1,2]) == [-2,1,2]
    print dropWhile(lambda x: x>0, []) == []
```

SOLUTION

```
def dropWhile(f, L):
    if L == []:
        return []
    elif f(L[0]):
        return dropWhile(f, L[1:])
    else:
        return L
```

Here's another version:

```
def dropWhile(f, L):
    if L == []:
        return []
    elif not f(L[0]):
        return L
    else:
        return dropWhile(f, L[1:])
```

Rubric: 3 syntactically reasonable attempt + 4 base case + 4 case where f is true + 4 case where f is false

Question 5 *(10 points)* **Assess:** [state]

Using the definitions below, write down a trace of the call power(3,4). Just show the function calls with argument values, and indicate what is the final value returned.

```
def power(x,y):
    '''Compute x**y assuming y is an integer >=0 and x a number.'''

    def pow(x,y,accum):
        if y==0: return accum
        else: return pow(x,y-1,x*accum)

    return pow(x,y,1)
```

SOLUTION

```
power(3,4)
  pow(3, 4, 1)
    pow(3, 3, 3)
      pow(3, 2, 9)
        pow(3, 1, 27)
           pow(3, 0, 81)
```

Rubric: 4 more than one correct call (with arguments) + 4 all and only the correct calls + 2 result is same as third parameter, or 81

Question 6 *(20 points)* **Assess:** [coding]

Complete the implementation of this function. Use recursion and the "use it or lose it" strategy.

```
def powerset(L):
    """
    The set of sub-lists of L.  The sub-lists
    should be in same order as in L, but the order
    of sub-lists is arbitrary.

    For example, powerset(["cat", "mouse", "mole"]) could
    return [[], ["cat"], ["mouse"], ["mole"], ["cat", "mouse"],
            ["cat", "mole"], ["mouse", "mole"], ["cat", "mouse", "mole"]]
    """
    if L==[]: return [[]]
    else:
        lose = ????
        use = map(lambda M: [ L[0] ] + M, ???? )
        return ????
```

You just need to figure out the parts marked ????, but please write down the entire "else" part of the code.

SOLUTION
Here's a solution:

```
lose = powerset(L[1:])
use = map(lambda x: [L[0]] + x, lose)
return lose + use
```

What about this code?

```
lose = powerset(L[1:])
use = map(lambda x: [L[0]] + x, powerset(L[1:]))
return lose + use
```

It works correctly so we gave it full credit. But it runs slower, because it re-computes `powerset(L[1:])` instead of reusing it.

Rubric: 2 for syntax ok + 6 for lose + 6 for use + 6 for return

```
# SOLUTION - cs115 Fall 2014, test2 Questions 7 and 8

#################################################################
# Name:
# Date:
# Pledge:
#################################################################


##############
# Question 7a
# Create a dictionary INT_TO_HEX_DIGIT than maps integers
# to their corresponding hexadecimal digits.
# 0 -> '0'
# 1 -> '1'
# ...
# 9 -> '9'
```

```
# 10 -> 'A'
# 11 -> 'B'
# ...
# 15 -> 'F'
##############

INT_TO_HEX_DIGIT = {  0 : '0',
                      1 : '1',
                      2 : '2',
                      3 : '3',
                      4 : '4',
                      5 : '5',
                      6 : '6',
                      7 : '7',
                      8 : '8',
                      9 : '9',
                     10 : 'A',
                     11 : 'B',
                     12 : 'C',
                     13 : 'D',
                     14 : 'E',
                     15 : 'F' }

##############
# Question 7b
# Create a dictionary HEX_TO_INT_NUMBER than maps hexadecimal
# digits to their corresponding integer values.
# '0' -> 0
# '1' -> 1
# ...
# '9' -> 9
# 'A' -> 10
# 'B' -> 11
# ...
# 'F' -> 15
##############

HEX_TO_INT_NUMBER = { '0' : 0,
                      '1' : 1,
                      '2' : 2,
                      '3' : 3,
                      '4' : 4,
                      '5' : 5,
                      '6' : 6,
                      '7' : 7,
                      '8' : 8,
                      '9' : 9,
                      'A' : 10,
                      'B' : 11,
                      'C' : 12,
                      'D' : 13,
                      'E' : 14,
                      'F' : 15 }
```

```
##############
# Question 8
# Implement the functions integer_to_hex_helper and hex_to_integer.
# Use recursion and use the dictionaries you defined for Question 7.
#
# Hint: read everything before starting to work.
# The test code provides examples of what the functions should do.
##############

def integer_to_hex(num):
    '''Assume num is an integer.
       Convert it to a hexadecimal (base 16) string.'''

    def integer_to_hex_helper(num):
        '''Assume num is an integer.  Convert it to hexidecimal, without any
           leading zeros.  In particular, return the empty string for 0.'''
        if num == 0:
            return ''
        return integer_to_hex_helper(num / 16) + INT_TO_HEX_DIGIT[num % 16]

    # We are calling your helper function to ensure there is no leading 0
    # in the return value.
    return '0' if num == 0 else integer_to_hex_helper(num)

def hex_to_integer(hex):
    '''Assume hex is a string that represents a number in base 16.
    Convert it to an integer.'''
    if hex == '':
        return 0
    return 16 * hex_to_integer(hex[:-1]) + HEX_TO_INT_NUMBER[hex[-1]]


##############
# Test code.
# Notice that hexidecimals are strings of characters, whereas integers are not.
##############

def test():
    '''Prints a success message, or assertion error if a test fails.'''
    assert integer_to_hex(0) == '0'
    assert integer_to_hex(7) == '7'
    assert integer_to_hex(11) == 'B'
    assert integer_to_hex(20) == '14'
    assert integer_to_hex(255) == 'FF'   # notice 255 is decimal notation for a integer
    assert integer_to_hex(1000) == '3E8'
    assert hex_to_integer('0') == 0
    assert hex_to_integer('A') == 10
    assert hex_to_integer('48321') == 295713
    assert hex_to_integer('DEADBEEF') == 3735928559
    assert hex_to_integer(integer_to_hex(10)) == 10
    assert hex_to_integer(integer_to_hex(0)) == 0
    assert hex_to_integer(integer_to_hex(12345)) == 12345
    print "Tests were successful."
```

```
# test_composition()
```

Rubric: Q7: 5 points for each correct dictionary **Rubric:** Q8: for each of the two functions, 2 points if compiles + 3 points for correct base case + 5 points for correct recursive call