Midterm Practice Solutions

CS 111

1.

a. 'C'

b. 'C'

c. 'olumbus'

d. 's'

e. 'uloC'

f. 'lmu'

2.

a. 3

b. [3]

c. 5

d. [4, 7]

e. [3, 2]

f. [-1, 5]

3.

a. True

b. [1, 2, 3, 13, 33, 44, 55]

c. [3, 3, 3, 3, 3, 3]

d. [3, 7, 11, 15]

e. [4, 6, 8]

f. [2, 8]

# Exercise 4

def count\_ones(s):

"""Return the number of 1s in s.

Parameters:

s -- a string of 0s and 1s

"""

if s == '':

return 0

else:

num\_ones\_in\_rest = count\_ones(s[1:])

if s[0] == '1':

return num\_ones\_in\_rest + 1

else:

return num\_ones\_in\_rest

# Exercise 5

def swap\_bits(s):

"""Return a new string where all of

the bits of s have been swapped.

Parameters:

s -- a string of 0s and 1s

"""

if s == '':

return ''

else:

swapped\_rest = swap\_bits(s[1:])

if s[0] == '1':

return '0' + swapped\_rest

else:

return '1' + swapped\_rest

# Exercise 6

def num\_divisors(n):

"""Return the number of the divisors of n.

Parameters:

n -- a nonnegative integer

"""

divisors\_lst = [x for x in range(1, n + 1) if n % x == 0]

return len(divisors\_lst)

# Exercise 7 (using a list comprehension)

def most\_divisors(lst):

"""Return the number in lst that has the

most divisors using a list comprehension

on a list of lists.

Parameters:

lst -- a list of nonnegative integers

"""

num\_divisors\_lst = [[num\_divisors(x), x] for x in lst]

max\_pair = max(num\_divisors\_lst)

return max\_pair[1]

# Exercise 7 (using recursion)

def most\_divisors2(lst):

"""Return the number in lst that has the

most divisors using recursion.

Parameters:

lst -- a list of nonnegative integers

"""

if len(lst) == 1:

return lst[0]

else:

most\_divisors\_in\_rest = most\_divisors2(lst[1:])

if num\_divisors(lst[0]) > num\_divisors(most\_divisors\_in\_rest):

return lst[0]

else:

return most\_divisors\_in\_rest

# Exercise 8

def count\_transitions(s):

"""Return the number of transitions from a '0'

to a '1' and vice versa in s.

Parameters:

s -- a string of 0s and 1s

"""

if len(s) <= 1:

return 0

else:

num\_transitions\_in\_rest = count\_transitions(s[1:])

if s[0] != s[1]:

return num\_transitions\_in\_rest + 1

else:

return num\_transitions\_in\_rest

# Exercise 9 (using a list comprehension)

def longest\_string(lst):

"""Return the longest string in lst using

a list comprehension on a list of lists.

Parameters:

lst -- a list of strings

"""

longest\_string\_lst = [[len(s), s] for s in lst]

longest\_pair = max(longest\_string\_lst)

return longest\_pair[1]

# Exercise 9 (using recursion)

def longest\_string2(lst):

"""Return the longest string in lst using recursion.

Parameters:

lst -- a list of strings

"""

if len(lst) == 1:

return lst[0]

else:

longest\_string\_in\_rest = longest\_string2(lst[1:])

if len(lst[0]) > len(longest\_string\_in\_rest):

return lst[0]

else:

return longest\_string\_in\_rest

# Exercise 10

def cycle(s, n):

"""Return the string that results from cycling

the last character of s to the beginning n times.

Parameters:

s -- a string of 0s and 1s

n -- a nonnegative integer

"""

if s == '':

return ''

elif n == 0:

return s

else:

new\_s = s[-1] + s[:-1]

return cycle(new\_s, n - 1)

11.

in dog, x is 3

in cat, y is 2

in rat, x is 4

in cat, x is 11

in cat, y is 5

in rat, x is 10

in cat, x is 23

in dog, y is 34

at this level, y is 34

12.

x is 3

x is 2

x is 1

x is 0

p is 4

p is 2

p is 4

y is 4