CS115 Test 2 Sample

Name:
Pledge:

Closed book: no textbook, no electronic devices, one sheet of paper with handwritten notes. *Read carefully before answering!* Write your answers on this test paper. Also hand in your note sheet, with your name on it.

Question 1 *(20 points)*

(a) What is the binary representation of forty seven (i.e., 47_{10})? Write it using exactly 8 bits.

(b) Compute the sum of these two binary numbers, in binary, showing the carry bits.

$$\begin{array}{rr} & 101101 \\ + & 100111 \\ \hline \end{array}$$

(c) Using two's complement with exactly 8 bits, what is the binary representation of negative twenty three? (i.e. -23_{10}).

(d) Using two's complement with 7 bits, what is the largest positive integer that can be represented? What is the smallest (most negative)? Write your answers in base 10.

Question 2 *(15 points)*

Here is a table that defines a boolean function f with three inputs.

x	y	z	$f(x, y, z)$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Complete the following Python implementation of the function, using a single return expression. Your expression should be based on the minterm expansion principle. Use the built-in Python functions "and", "or", "not".

```
def f(x,y,z):


    return ???
```

Question 3 *(15 points)* Using assert statements, implement the following function so that it tests at least four different cases for function `f` from Question 2.

```
def test_f():
```

Question 4 *(15 points)* What gets printed by this Python code?

```
mem = {('', 'st'): 0, ('', 't'): 0, ('a', ''): 0, ('a', 't'): 0, ('a', 'st'): 0, \
       ('ea', 'est'): 1, ('tea', 'test'): 2}

mem[('a', 't')] = 8

print( ('e', 't') in mem )

print( mem[('ea', 'est')] )

print( mem[('a', 't')] )
```

Question 5 *(15 points)*

Complete the following function, using recursion on `L`. That means you can only 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 L 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, []) == [] )
```

Question 6 *(20 points)*

This is about memoization of a function called `LAS` which applies to a string `S` and a letter `ltr`. It returns a *longest ascending subsequence of `S` whose elements are all greater than `ltr`.* Reminder: characters are compared according to their numerical encoding, which agrees with alphabetical order in the case of letters. For example, `'a' < 'b'` is true, and `max('a','b')` returns `'b'`. Study these examples, which are all true:

```
LAS("", 'b') == ""
LAS("bcd", 'd') == ""
LAS("bcd", 'a') == "bcd"
LAS("bbccdd", 'a') == "bcd"
```

The last example shows that which shows this is about *subsequences*, not sub-segments. Note that for `LAS("bbccdd", 'a')`, the specification allows both of these: `"bcd"` `"bdf"`

The code below works correctly, if we delete the ??? parts. *YOUR TASK:* replace each ??? with some code, so that it uses a dictionary as a memo table. For keys, the dictionary should use pairs `(S,ltr)`. Hint: the first ??? should initialize the dictionary.

```
def LAS(S, ltr):
      '''Assume S is string of letters and ltr is a letter.
        Return a longest subsequence of S that is increasing and
        whose elements are all greater than ltr.'''

         ???

         ???


         if (S,ltr) in Tab: return Tab[(S,ltr)]

         if S == "":
             result = ""
         elif S[0] <= ltr:
             result = LAS(S[1:], ltr)
         else:
             use = S[0] + LAS(S[1:], max(ltr, S[0]))
             lose = LAS(S[1:], ltr)
             if len(use) >= len(lose): result = use
             else: result = lose

         ???


         return result
```