CS115 Final Sample —SOLUTION GUIDE—

Question 1 *(5 points)* Using two's complement with exactly 8 bits, what is the binary representation of negative forty two? (i.e. -42_{10}).

SOLUTION 11010110
Because 42 is 00101010, the one's complement of which is 11010101

Question 2 *(5 points)* What is the value of this expression?

```
list(filter(lambda x: 10 <= x and x <= 40, map(lambda x: x * 5, [1,4,6,9])))
```

SOLUTION [20, 30]

Question 3 *(5 points)* What is the value of this expression?

```
list(map(lambda x: x+1, filter(lambda x: 1 <= x and x < 9, [1,3,7,9])))
```

SOLUTION [2, 4, 8]

Question 4 *(5 points)* Implement the following, using `map` and `lambda` (not recursion). Hint: all it needs is a return statement.

```
def exclaim(strs):
    ''' Assume strs is a list of strings.
        Return a new list of the same strings but with ! added at the end.'''
```

For example, `exclaim(["I", "can", "do", "this"])` is `["I!", "can!", "do!", "this!"]`.

SOLUTION

```
  return list(map(lambda x: x+'!', strs))
```

Question 5 *(5 points)* Implement the following function, using `filter` and `lambda` (not recursion).

```
def trimLimit(L, lower, upper):
    ''' Assume L is a list of number. Return the list of elements of L
        that are between the numbers lower and upper (inclusive).'''
```

For example, `trimLimit([2,19,5,13,12], 5, 20)` returns `[19,5,13,12]`. So does `trimLimit([2,19,5,13,12],5,19)`.

SOLUTION

```
def trimLimitSol(L, lower, upper):
    return list(filter(lambda x: lower <= x <= upper, L))
```

As the example shows, "between" means we should use `<=` not `<`.

Question 6 *(5 points)* Implement the function `trimLimit` from the preceding question, this time using recursion on `L` and not using `filter`.

SOLUTION

```
def trimLimRec(L, lower, upper):
    if L == []:
        return []
    elif lower <= L[0] <= upper:
        return [L[0]] + trimLimRec(L[1:], lower, upper)
    else:
        return trimLimRec(L[1:], lower, upper)
```

Question 7 *(5 points)* Here is the recursive definition of the fib function, in math notation. Assume n is a non-negative integer.

$$\begin{array}{rcll} fib(n) & = & n & \text{if } n \leq 1 \\ & = & fib(n-1) + fib(n-2) & \text{otherwise} \end{array}$$

Write it as a Python function definition. No docstring required.

SOLUTION

```
def fib(n):
    if n==0 or n==1:
        return n
    else:
        return fib(n-1) + fib(n-2)
```

Question 8 *(5 points)* Have a look at this code.

```
def iFib(n):
    '''Imperative implementation of fib.'''
    if n==0 or n==1: return n
    else:
        prev = 0
        cur = 1
        i = 1
        while i!=n:
            cur = cur + prev
            prev = cur - prev
            i += 1

    return cur
```

Fill in the table below, as a loop trace for the call `iFib(4)`. The first row already shows the initial values. Add one row for each iteration of the loop, that shows their values after that iteration. So the last row is the final values, just before `cur` is returned.

```
prev cur  i
------------
0    1    1
```

SOLUTION

```
prev cur  i
------------
0    1    1
1    1    2
1    2    3
2    3    4
```

Question 9 *(5 points)* Add two assignment statements at the places marked "???" so that this function works correctly.

```
def binsearch(L, i, x):
    '''Assuming L[0:i] is sorted and 0 <= i <= len(L),
       return j such that 0 <= j <= i and L[0:j] <= x < L[j:i].'''
    j = 0
    hi = i
    # invariant: L[0:j] <= x < L[hi:i] and j <= hi
    while j != hi:
        mid = (hi + j) // 2
        if x < L[mid]:
```

```
            ???

        else:

            ???

    return j
```

SOLUTION First is `hi = mid`, second is `j = mid + 1`

Question 10 *(10 points)* What gets printed by this code? Hint: you might want to draw a diagram to figure it out.

```
L = ["winter", "break"]
M = ["fake","news"]
N = [L, M, L]
P = list(N)
P[2][0] = "fun"
print(N)
```

SOLUTION

```
[['fun', 'break'], ['fake', 'news'], ['fun', 'break']]
```

Question 11 *(10 points)* Fill in the missing parts (marked ???) to complete the following function using recursion.

```
def subset(capacity, items):
    '''Given a suitcase capacity and a list of items consisting of positive
       numbers, returns a number indicating the largest sum that can be made
       from a subset of the items without exceeding the capacity.  Items
       cannot be re-used.'''

    if capacity <= 0 or items == []:
        return 0
    elif items[0] > capacity:

        return ???


    else:


        lose = ???


        use = ???


        return max(use,lose)
```

SOLUTION

```
        return subset(capacity, items[1:])

        lose = subset(capacity, items[1:])

        use = items[0] + subset(capacity - items[0], items[1:])
```

Question 12 *(10 points)* Implement the following function, using a loop and not recursion.

```
def locationOfMax(L):
    '''Assume L is a non-empty list of numbers.  Return an index i
       such that L[i] is at least as large as every element of L.'''
```

For example, `locOfMax([3,1,9,5,0])` returns `2` and `locationOfMax([3])` returns `0`.

SOLUTION

Here is one solution.

```
def locOfMax(L):
    '''Assume L is a non-empty list of numbers.  Return an index i
       such that L[i] is at least as large as every element of L.'''
    loc = 0
    for i in range(1,len(L)):
        if L[i] > L[loc]:
            loc = i
    return loc
```

Question 13 *(5 points)* Below is the recursive definition of the LCS function that computes the length of the longest common subsequence of two strings. *Show the trace of function calls starting from* `LCS("tap","trap")`.

```
def LCS(S1, S2):
    ''' Length of the longest common subsequence of S1 and S2.'''

    if S1 == "" or S2 == "": return 0
    elif S1[0] == S2[0]:
        return 1 + LCS(S1[1:], S2[1:])
    else:
        chopS1 = LCS(S1[1:], S2)
        chopS2 = LCS(S1, S2[1:])
        answer = max(chopS1, chopS2)
        return answer
```

SOLUTION

```
LCS("tap","trap")
   LCS("ap", "rap")
      LCS("p", "rap")
         LCS("", "rap")
         LCS("p", "ap")
            LCS("", "ap")
            LCS("p", "p")
               LCS("", "")
      LCS("ap", "ap")
         LCS("p", "p")
            LCS("", "")
```

Question 14 *(10 points)* What gets printed by the following code?

```
class Person:
    def __init__(self, first, last):
        self.firstName = first
        self.lastName = last

    def asleep(self, time):
        return 0 <= time <= 7

    def __str__(self):
        return self.firstName + " " + self.lastName
```

```
class Student(Person):
    def __init__(self, first, last, gpa):
        Person.__init__(self, first, last)
        self.gpa = gpa

    def asleep(self, time):
        return 3 <= time <= 11

    def __str__(self):
        return Person.__str__(self) + ", " + " GPA=" + str(self.gpa)

class SITstudent(Student):
    def __init__(self, first, last):
        Student.__init__(self, first, last, 0)
        self.grades = []

    def __str__(self):
        return Person.__str__(self)

    def status(self, time):
        if self.asleep(time):
            return str(self) + " is asleep now."
        return str(self) + " is gaming or studying."

    def addGrade(self,score):
        '''Add score to list of grades (assume score is number in range 0..100).'''
        self.grades += [score]
        self.gpa = sum(self.grades)/len(self.grades)

    def getGrades(self): return self.grades


# print someone's status at time 10
someone = SITstudent("Jean", "Zu")
print(someone.status(10))
```

SOLUTION `Jean Zu is asleep now.`

Question 15 *(10 points)* In the context of the classes in the previous question, what gets printed by the following code?

```
alice = SITstudent("Alice", "Cleverness")
alice.addGrade(70)
alice.addGrade(80)
print("Alice's grades " + str(alice.getGrades()))
agrad = alice.getGrades()
agrad[0] = 100
print("Alice's grades now " + str(alice.getGrades()))
```

SOLUTION

```
Alice's grades [70, 80]
Alice's grades now [100, 80]
```