Question 1. [10 MARKS]

Part (a) [2 MARKS] Complete the following function according to its docstring description. Your answer must be a single line of code.

```
def add_to_month(month, n):
    '''Month is an integer between 1 and 12 inclusive; n is an integer
    that is at least 0 and represents a number of months.  Return an integer
    between 1 and 12 inclusive that represents what month it will be if we add
    n to month.'''
```

Solution: This answer has the main idea, but doesn't work in some cases:

```
return (month + n) % 12
```

The problem is that `% n` can give you any number from 0 to n-1. Sometimes that's what you want. For instance, for the assignment 2 "shift_message" function, many students created either a string or a list containing the letters of the alphabet, and added to a letter's index in order to shift it. You could use `% 26` in that case, because you wanted a result between 0 (the index of 'a') and 25 (the index of 'z'). But here we want a month number, which must be between 1 and 12, not 0 and 11. The following works:

```
return 1 + (month + n - 1) % 12
```

That's pretty tricky. We gave nearly full marks for the first answer above.

Part (b) [3 MARKS] Complete the following function according to its docstring description. Again, your answer must be a single line of code.

```
def rotate(L, n):
    '''L is a list and n is an integer.  0 <= n <= len(L).  Return a new list which is
    L rotated to the left by n positions.
    For example, rotate([1, 2, 3, 4, 5, 6], 2) returns [3, 4, 5, 6, 1, 2].'''
```

Solution:

```
    return L[n:] + L[:n]
```

Part (c) [5 MARKS] Complete the following function according to its docstring description.

```
def short_strings(L, n):
    '''Return a new list that contains all the elements of list L whose
    length is less than n.'''
```

Solution:

```
    new = []
    for item in L:
        if len(item) < n:
            new.append(item)
    return new
```

Question 2. [6 marks]

Here is a mystery program:

```
def mystery(s):
    ans = []
    while s != "":
        if len(s) == 1:
            ans.append(s)
            s = ""
        else:
            ans.append(s[0] + s[1])
            s = s[2:]
    print "ans is", ans
    return ans

if __name__ == "__main__":
    var1 = "Pooh-bear!"
    var2 = mystery(var1)
    print "var1 is", var1
    print "var2 is", var2
```

What is the output of this program?

Solution:

```
ans is ['Po', 'oh', '-b', 'ea', 'r!']
var1 is Pooh-bear!
var2 is ['Po', 'oh', '-b', 'ea', 'r!']
```

Question 3. [8 marks]

Part (a) [4 marks] What is the output of the following program?

```
def loopy(s):
    new = ""
    i = 0
    while i < len(s):
        print "Outer", i, new
        if s[i] == "-":
            while s[i] == "-":
                print "Inner", i, new
                i = i + 1
            new = new + "-"
        else:
            new = new + s[i]
            i = i + 1
    print "final result", new

if __name__ == "__main__":
    loopy("a---ha")
```

Solution:

```
Outer 0
Outer 1 a
Inner 1 a
Inner 2 a
Inner 3 a
Outer 4 a-
Outer 5 a-h
final result a-ha
```

Part (b) [4 marks] Write a docstring for `loopy`. (Pretend the print statements aren't there — they were temporary.)

Solution: Without a return statement, this function has no effect at all! But this was a typo, not a trick question — the function was intended to say "`return new`" at the end. So full marks were given for saying either that the function has no effect, or:

```
'''Return a new string that is the same as string s, but with duplicate consecutive hyphens
each replaced by a single hyphen.'''
```