Section 1: (50 points) Each example consists of Python code, followed by questions and places to fill in answers. Please read all questions carefully and answer them as precisely as possible.

Assume that there are no bugs in the code that will make the system crash, although the code may not solve a problem perfectly. If you find anything that you think is a bug, there is either a typo (and I should fix it for everyone) or you are mistaken.

Sample Question A:

```python
output = '1'+1'
```

**Question:** What is the value of the variable `output`?

**Answer:** '11'

**Note:** Attention to detail is important. The quotes indicate that it is a string. Partial credit is possible. For example, leaving out the quotes would have lost just a little bit, but answering 2, would have resulted in an incorrect answer.

**Question 1**

```python
def plutify(number1, number2):
    output = (number1 ** number2)/number2
    return(output)

answer = plutify(3, 4)
```

**Question:** What is the value of the global variable `answer` after the above code is executed?
Question 2

def draw_martian():
    for num in range(7):
        print('', end='')
    print('*')
    for num in range(5):
        print('', end='')
    print('*<O>*')
    for num in range(7):
        print('', end='')
    print('*')
    for num in range(7):
        print('', end='')
    print('*')
    for num in range(7):
        print('', end='')
    print('*')
    for num in range(7):
        print('', end='')
    print()  
    for num in range(8):
        print('* ', end='')
    print()  
    for num in range(7):
        print('', end='')
    print()  
    for num in range(7):
        print('', end='')
    print('*')
    for num in range(6):
        print('', end='')
    print('* *')
    for num in range(6):
        print('', end='')
    print('*')
    for num in range(3):
        print('', end='')
    for num in range(6):
        print('', end='')
    print()  

draw_martian()

Draw approximately what the above code causes to print out. A 14 by 14 grid is provided, to make this easier.
Question 3

def apply_commission(starting_amount, commission, rounds):
    total = starting_amount
    for number in range(rounds):
        total = total-(total*commission)
    return(total)

output = apply_commission(400, .50, 3)

Question: What is the value of the global variable output after the above code is executed?
def yes_or_no(question):
    user_answer = 'nothing'
    while (user_answer != 'yes') and (user_answer != 'no'):
        if user_answer != 'nothing':
            print('Improper Input: only type "yes" or "no"')
            print(question)
            user_answer = input('Please answer "yes" or "no". ')
        if user_answer == 'yes':
            return(True)
        else:
            return(False)

def mad_libs():
    noun1 = input('Give me a noun. ')
    number1 = input('Give me a number. ')
    noun2 = input('Give me a noun. ')
    noun3 = input('Give me a noun. ')
    adj1 = input('Give me an adjective. ')
    verb1 = input('Give me a present tense verb ending in "s". ')
    plural1 = input('Give me a plural noun. ')
    noun4 = input('Give me a noun. ')
    adj2 = input('Give me an adjective. ')
    print("There's been a lot of noise made about Zealandia as a new", noun1+".")
    print("And while more than",number1, "percent of this", \\
    noun2, noun3,'is',adj1+",")
    print("scientists believe that it",verb1,plural1,"
    'to be considered the worlds eighth",noun4+".")
    print("But there are some who disagree, asserting that Zealandia doesn't \
    fit the bill as a",adj2,noun1+".")

mad_libs()

# Based on paragraph from news story:
# http://www.inquisitr.com/3994334/is-zealandia-a-continent
# -scientists-see-it-as-one-but-not-everyone-agrees/
# There's been a lot of noise made about Zealandia as a new
# continent. And while more than 90 percent of this land mass is
# underwater, scientists believe that it fulfills the criteria to be
# considered the worlds eighth continent. But there are some who
# disagree, asserting that Zealandia doesn't fit the bill as a
# conventional continent.

Question: What is printed out if the user answers the questions posed by the calls to the input function with the following answers in the following order:

1. chicken
2. 57
3. bean
4. moss
5. wooly
6. flies
7. noses
8. toaster
9. intelligent
Section 2 (50 points): Answer 2 of the 3 questions in this section. For each question, you do, write a Python program as specified. If you choose to answer all 3 questions (only do this if you really have enough time), please indicate which ones you would like to count for the test.

Question 5: Write a calculator program that allows the user to combine pairs of two numbers together with plus, times, minus or divide. Use a while loop that exits when the user says that they are done combining numbers. The program should keep asking the user to choose two numbers and an operation to combine them with. The program should print out the result of combining the numbers together. A sample execution of the program follows.

```python
>>> calculator()
Would you like to combine 2 numbers? yes
Give me a number 5
Give me another number 5
choose: plus, times, minus, or divide. times
The answer is: 25.0
Would you like to combine 2 numbers? yes
Give me a number 7
Give me another number 3
choose: plus, times, minus, or divide. divide
The answer is: 2.3333333333333335
Would you like to combine 2 numbers? yes
Give me a number 5
Give me another number 2
choose: plus, times, minus, or divide. minus
The answer is: 3.0
Would you like to combine 2 numbers? no
```
Question 6: Write a function `modulus_table` that takes one parameter, `maximum` (an integer) and calculates modulus for all combinations of numbers from 1 to maximum. A sample execution of the program follows. Remember that the modulus operator is the percent sign (%).

```python
>>> modulus_table(5)
1 modulus 1 equals 0
1 modulus 2 equals 1
1 modulus 3 equals 1
1 modulus 4 equals 1
1 modulus 5 equals 1
2 modulus 1 equals 0
2 modulus 2 equals 0
2 modulus 3 equals 2
2 modulus 4 equals 2
2 modulus 5 equals 2
3 modulus 1 equals 0
3 modulus 2 equals 1
3 modulus 3 equals 0
3 modulus 4 equals 3
3 modulus 5 equals 3
4 modulus 1 equals 0
4 modulus 2 equals 0
4 modulus 3 equals 1
4 modulus 4 equals 0
4 modulus 5 equals 4
5 modulus 1 equals 0
5 modulus 2 equals 1
5 modulus 3 equals 2
5 modulus 4 equals 1
5 modulus 5 equals 0
```
Question 7: Implement a decision tree based program for classifying writing implements. Use the flow chart below to write the program. Use the variables below rather than writing out all the questions – this should save time. The program asks questions in order to set the following variables (listed in allcaps in the flowchart): COLOR, IMPLEMENT, SUBTYPE and BIRD. While COLOR and IMPLEMENT are always set in the course of the program, SUBTYPE and BIRD are initialized to the empty string ””. These will be the default values if the program does not change their settings. After asking questions, the program should print out: “You have a COLOR BIRD SUBTYPE IMPLEMENT”, replacing the variables with their respective values. If a variable is set to the empty string, it should not be printed out. There should also be no extra spaces between words (caused by printing spaces between an empty string and something else).

Note that the questions in the flowchart are numbered, e.g., there is a 1 under the question corresponding to the variable quest1. This should help you to quickly associate these boxes with the variable names.

quest1 = ‘Does your writing implement use ink?’
quest2 = ‘Is your writing implement a feather?’
quest3 = ‘What kind of bird?’
quest4 = ‘Does it have a metal ball at the tip?’
quest5 = ‘Does it have a felt tip’
quest6 = ‘Does it contain a graphite cylinder?’
quest7 = ‘Does it have mechanical parts?’
quest8 = ‘Is it made out of a wax?’
quest9 = ‘Is it made of calcite?’
Initialize variables SUBTYPE and BIRD to the empty string. Set COLOR to the answer of "What color?"

Does your writing implement use ink?

Yes → Set IMPLEMENT to 'pen'
No → Is your writing implement a feather

Yes → Set SUBTYPE to 'quill'
No → Does your writing implement a feather

Yes → Print out "You have a COLOR BIRD SUBTYPE IMPLEMENT"
No → Does it have a metal ball at the tip?

Yes → Set SUBTYPE to 'ballpoint'
No → Does it have a felt tip?

Yes → Set SUBTYPE to 'felt tip'
No → Does it contain a graphite cylinder?

Yes → Set SUBTYPE to 'mechanical'
No → Does it have mechanical parts?

Yes → Set SUBTYPE to 'mechanical'
No → Is it made of wax?

Yes → Set IMPLEMENT to 'pencil'
No → Set IMPLEMENT to 'unknown writing implement'

Does it made of calcite?

Yes → Set IMPLEMENT to 'chalk'
No → Set IMPLEMENT to 'unknown writing implement'
Basic Stuff to Look Up for the Test

1. Some Basics

- **return(X)** causes the current function to exit and cause the expression represented by the function call to evaluate as X. For example given the following steps, the value of `output` would be 5:
  ```python
def add(num1, num2):
    return(num1+num2)
output = add(2, 3)
```
- **print(X)** prints X to the screen. This is only for the benefit of the user. It is not useful for having programs interact.
- The parameters of a function are the local variables inside of the parentheses in the function definition. They are useful when you have functions call functions.
- **input(prompt)** is used to ask a human being a question so that a program can interact with a human being. This is useful when you want a human being to enter information interactively. input statements should be used only when human interaction is appropriate. input statements return a string corresponding to what the user typed in. It may be necessary to convert this string to some other data type, e.g., an integer (with `int`) or a float (with `float`).
- The operator `+` will add two numbers or concatenate two strings
- The operator `*` will multiple two numbers or print a string some number of times.
- The operator `**` will represent exponents, e.g., `5**2 == 25`

2. Division and Modulus

- `5 // 2 == 2`
- `5/2 == 2.5`
- `5%2 == 1`

3. More Math

- `round(4.5) == 4, round(4.6) == 5, round(-4.5) == -4, round(-4.6) == -5`
- `round(2/3,2) == .67` ## The second argument of round indicates number of decimal places. The default is to round to the nearest whole number.
- `math.floor(4.9) == 4, math.floor(-4.9) == 5`
- `math.ceil(4.9) == 5, math.ceil(-4.9) == -4`
- `math.trunc(4.9) == 4, math.trunc(-4.9) == -4`
- `math.pi == 3.141592653589793` – a variable for the value of pi

4. sequences

- object made up of other objects in an order
- the function `len(sequence)` returns the number of items in the sequence
- the operator `in` tests for membership in sequence, e.g., (’a’ in ’abc’) would have the value True.
- sequences are used in for loops (see below)
- ranges
  - `range(5)` is approximately equivalent to [0,1,2,3,4]
  - `range(1,5)` is approximately equivalent to [1,2,3,4]
- Strings
  - an empty string has zero characters ”
strings are sequences of characters, e.g., 'Hello World!' consists of the items ['H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '!']

5. print
   - **sep** – separator between items
   - **end** – printed at the end of print statement

6. for loops
   - First Line: *for VARIABLE in SEQUENCE:*
   - VARIABLE is set to each item in the sequence one at a time
   - The Indented body repeats once for each item in sequence (for each setting of VARIABLE).
   - It is common to exit a loop of any kind by using a *return* to exit the function.
   - It is common to initialize a variable outside a loop (called an accumulator) that then gets incremented inside the loop.

7. if statements
   - the first line of an *if* statement consists of *if* **BOOLEAN-EXPRESSION:**
   - the body of text indented under the first line is executed if the **BOOLEAN-EXPRESSION** evaluates to True
   - the *if* statement can be followed by optional *elif* statements of the same form, except that the first line begins with *elif*. Each *elif* statement is only evaluated if the **BOOLEAN expressions** in the *if* and *elif* statements leading up to this one are False.
   - The block of *if* and optional *elif* statements can end with an optional *else* statement. The first line is simply *else*: The body of text under *else* executes if the **BOOLEAN expressions** for all previous *if* and *elif* statements in the sequence evaluate to False.

8. logical operators
   - X **and** Y returns True only if both X and Y are True
   - X **or** Y returns True only if X is True, Y is True or both are True
   - X **in** Sequence returns True if X is in a member of a sequence, e.g., 'a' in 'abcdefg' would return True
   - X **==** Y returns True if X and Y are the same
   - X **!=** Y returns True if X and Y are the different
   - X **<** Y returns True if X is less than Y
   - X **>** Y returns True if X is greater than Y
   - X **<=** Y returns True if X is less than or equal to Y
   - X **>=** Y returns True if X is greater than or equal to
   - **Not** X returns True if X is False