Name: ________________________________
Net ID ________________________

There are 2 sections, the first section is worth 50 points and the second section is worth 50 points, for a total of 100.
It is essential that you **PUT YOUR NAME AND NET ID ON ALL TEST MATERIALS**. It can be difficult to identify the author of an unsigned test and it would be better to avoid this problem.

**There is a GLOSSARY OF TERMS at the end of the test. Please feel free to look up some of the basics in this glossary. I will also answer any reasonable look-up style questions as I am more interested in your ability to reason than your ability to memorize.**

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 pretend_function(number,pretend_number):
    output = number+pretend_number
    print('So far the output is: ',output)
    output = output*pretend_number
    print('So far the output is: ',output)
    output = output/pretend_number
    print('So far the output is: ',output)
    output = output-pretend_number
    print('So far the output is: ',output)
    ## round returns the closest whole number
    output = round(output)
    return(output)

answer = pretend_function(57,50)
```

**Question 1a:** What is the value of the global variable `answer` after the code executes?

**Question 1b:** What text print out during the execution of the above code?
Question 2

def print_pet():
    print(' ˆ'+(' '*4+'ˆ')+(' '*6)+'*')
    print('<O> <O>'+('_' *5)+')')
    print((' '*3)+'[]'+(' '*2)+'/\'+(' '*3)+'/\')
    print((' '*6)+'/'+(' '*2)+'\'+(' '*1)+'/'+(' '*2)+'\')
    print_pet()

print_pet()

Please note that \\ represents the single character \ because the backslash is an escape character, a character used in two character combinations to represent special characters: \n represents a newline character, \t represents a tab and \\ represents the backslash itself.

Question: Please draw (approximately) what would be printed out by executing the above code.
Question 3

def triangle_number(input_number):
    total = 0
    for number in range(1, input_number + 1):
        total = total + number
    return(total)

def square_minus_triangle(number1):
    square = number1 * number1  ### line 2
    triangle = triangle_number(number1)  ### line 3
    total = square - triangle
    return(total)

answer = square_minus_triangle(5)

Question 3a. What is the value of answer after the above code is executed?

Question 3b. During the execution of the function square_minus_triangle above, number1 is set to 5 by the function call. What is the value of the local variable square after the execution of the line tagged with the comment ### line 2?

Question 3c. During the same function call, what is the value of the local variable triangle after executing the line labeled ### line 3?
Question 4

```python
def guess_positive_integer(label, maximum):
    answer = int(input('Guess '+label+': '))
    while (answer < 1) or (answer > maximum):
        print('Your answer is not between 1 and',maximum)
        answer = int(input('Guess another'+label+': '))
    return(answer)

def guess_birthday(month,day):
    print('Guess month and day in numeric form.
    print('1 for January, 2 for February, ... 12 for December.
    month_guess = guess_positive_integer('month',12)
    day_guess = guess_positive_integer('day',31)
    guess_num = 1
    while (month != month_guess) or (day != day_guess):
        if month > month_guess:
            print('Try a later date')
        elif month < month_guess:
            print('Try an earlier date')
        elif day > day_guess:
            # assumes month is equal
            print('Try a later date')
        else:
            # assumes that day is less than day_guess
            print('Try an earlier date')
        month_guess = guess_positive_integer('month',12)
        day_guess = guess_positive_integer('day',31)
        guess_num = 1+ guess_num
    if guess_num == 1:
        print('Correct on 1st guess!')
    else:
        print('Correct on',guess_num,'guesses!')

guess_birthday(7,20)
```

The above function call begins a game in which the user is suppose to guess a month and a day (in numbers), such that the correct month is 7 and the correct day is 20. After each incorrect guess, the system instructs the user with helpful messages (see above). The game ends when the user makes a correct guess.

**Question 4a. What happens if a user enters 1 for the month and 35 for the day?**

**Question 4b. What happens if a user enters January for the month and 20 for the day?**

**Question 4c. What happens if a user enters 1 for the month and 20 for the day?**

**Question 4d. Give a plausible series of guesses (at least 5), combined with system responses, ending in the user correctly guessing the answer.**
Section 2 (50 points): Write 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, please indicate which ones you would like to count for the test.

Question 5: Write a program that will average as many numbers as are input by a user.

- The program should query the user using an input statement. The user should choose between: entering a number and typing the word stop. The input function should be inside a while loop.
- If the user types the word stop, this should cause the while loop to terminate. Otherwise, the number entered by the user should be converted to a float and included in the average.
- The main function of your program should return the average.
Question 6: Write a program that asks questions to a user about a stain on their clothing and classifies the stain based on the user’s answers. Implement your program based on the questions and the decision tree provided below. You can assume that the text of each question is encoded as a variable, e.g., you can use the variable name `question_1` to represent the string ‘What color is the stain’ and `question_2` to represent the string ‘Does the stain have a sharp edge around it’, etc.

1. What color is the stain?
2. Does the stain have a dark sharp edge around it?
3. Is the stain hard and crusty?
4. Does the stain smell like mustard?
5. Were you near a skunk recently?
**Question 7:** Write a program that will print a rectangle made up of alternating vertical stripes of asterisks and vertical bars. The specifications are as follows:

- The program should consist of a single function that takes two parameters: `height` and `width`.
- `Height` should indicate the number of rows that will be printed.
- `Width` should indicate the total number of characters in each row.
- In each row, the first character should be an asterisk, the second character should be a vertical bar, and so on. such that each odd numbered character is an asterisk and each even numbered character is a vertical bar.

For example the function call `striped_rectangle(5,10)` should produce the following output:

```
* | * | * | * | * |
* | * | * | * | * |
* | * | * | * | * |
* | * | * | * | * |
* | * | * | * | * |
```
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.

2. 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', '!']

3. Division and Modulus

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

4. print

- `sep` – separator between items
- `end` – printed at the end of print statement

5. 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.

6. *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.