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 add_two_ways(number1,number2,number3):
    print('Adding',number1,number2,'and',number3)
    sum_of_numbers = number1 + number2 + number3
    concatenation_of_numbers = str(number1)+str(number2)+str(number3)
    print('The answer is either',sum_of_numbers,'or',concatenation_of_numbers)
    return(sum_of_numbers)

answer = add_two_ways(5,7,5)
```

**Question 1a:** What prints out when the above code is executed?

**Question 1b:** What is the value of the global variable `answer` after the above code is executed?
Question 2

def print_Y_of_Ys():
    print('Y'+' '*7)+'Y'
    print(' '+'Y'+' '*5)+'Y'
    print(' '*2)+'Y'+' '*3)+'Y'
    print(' '*3)+'Y'+' '*1)+'Y'
    print(' '*3)+'Y'+' '*1)+'Y'
    print(' '*3)+'Y'+' '*1)+'Y'

print_Y_of_Ys()

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

```
__ __ __ __ __ __ __ __ __ __ __ __ __ __
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
```
def pile_of_asterisks(length):
    for num in range(length):
        space_num = length-(num+1)
        asterisk_num = (2*num)+1
        print((space_num*' ')+(asterisk_num*''))

def question_3():
    pile_of_asterisks(5)

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

__ __ __ __ __ __ __ __ __ __ __ __ __ __
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
|__|__|__|__|__|__|__|__|__|__|__|__|__|__|
def ask_question(question):
    answer = input(question + ' ')
    while (not ((answer == 'Yes') or (answer == 'yes') or
                (answer == 'y') or (answer == 'Y') or
                (answer == 'No') or (answer == 'no') or
                (answer == 'n') or (answer == 'N'))):
        answer = input("Please answer: yes or no")
    if (answer == 'Yes') or (answer == 'yes') or
        (answer == 'y') or (answer == 'Y'):
        return(True)
    else:
        return(False)

def suggest_a_fiction_book():
    question1 = 'Do you like fantasy and science fiction?'
    question2 = 'Do you care if the text is well-written?'
    question3 = 'Do you like character development?'
    question4 = 'Would you like it to be funny?'
    answer1 = ask_question(question1)
    answer2 = ask_question(question2)
    answer3 = ask_question(question3)
    answer4 = ask_question(question4)
    if answer1:
        if answer2:
            if answer4:
                print("Gulliver's Travels")
            else:
                print('Lord of the Rings')
        elif answer3:
            print('Harry Potter')
        else:
            print('Foundation Series')
    elif answer2 and answer3:
        print('Our Mutual Friend')
    elif answer4:
        print('Breakfast of Champions')
    else:
        print('Pippi Longstocking')

def question_4():
    suggest_a_fiction_book()

Question 4a: What will the program print out if the user answers 'yes' to questions 1 and 4, but 'no' to questions 2 and 3 (i.e., they want a funny fantasy or science fiction book, but they don’t care if it is well-written or has developed characters)?

Question 4b: What will the program print out if the user answers 'no' to question 1, but yes to the remaining questions (i.e., they want a non science fiction/fantasy book that is well-written, has good character development and is funny)?

Question 4c: What will the program print out if the user answers 'yes' to question 3, but 'no' to all the other questions (i.e., they want there to be character development, but they don’t want science fiction and do not care if it is well-written or funny)?
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 function called `word_repeater_and_counter`. It should take no parameter arguments. The function should solicit the user to input one word at a time or indicate they are done typing in words. The program should add each word the user types in to a string of words with spaces in between. The program should also count the number of words. When the user is done, the program should print out the sequence of words they typed in and then return the number of words. A sample interaction with such a function is provided below. In this example, an input function uses ‘Type next word or ”XXX” to indicate you are done.’ as a prompt. The user enters the word ‘blah’ 3 times and then enters XXX to indicate they are finished. The function prints `blah blah blah` and then returns the total number of words (not including ‘XXX’) that the user typed in.

```python
>>> word_repeater_and_counter()
Type next word or "XXX" to indicate you are done. blah
Type next word or "XXX" to indicate you are done. blah
Type next word or "XXX" to indicate you are done. blah
Type next word or "XXX" to indicate you are done. XXX
blah blah blah
3
```
Question 6: Write a decision-tree-based interactive fiction program, based on the flowchart in Figure 1 (on the next page). You do not have to type in the various strings into your program. Rather you can assume that your program has access to the variables defined below.

```
fluffy_question1 = 'Should she turn left?'
fluffy_question2 = 'Should she stay there and eat her sandwich?'
fluffy_question3 = 'Should she go home?'
fluffy_question4 = 'Should she pull the sword out?'

fluffy_bear1 = 'The fluffy bear was walking along the path'
fluffy_bear2 = '''She bumps into a tree, is knocked unconscious and wakes up 3 hours later.'''
fluffy_bear3 = '''She finds a clearing. There is a sword sticking out of a rock there.'''
fluffy_bear4 = '''The honey and rhubarb sandwich is really delicious. It is getting late.'''
fluffy_bear5 = 'It is getting late. So she goes home.'
fluffy_bear6 = '''It turns out to be a valuable artifact. She sells it and donates the proceeds to help bears everywhere.'''
```
The honey and rhubarb sandwich is delicious.

It is really getting late.

She finds a clearing, there is a sword sticking out of a rock there.

She bumps into a tree, is knocked unconscious and wakes up 3 hours later.

The fluffy bear was walking along the path.

Should she pull the sword out?
- Yes
  - It turns out to be a valuable artifact. She sells it and donates the proceeds to help bears everywhere.
- No
  - It is getting late. So she goes home.

Should she turn left?
- Yes
  - She finds a clearing, there is a sword sticking out of a rock there.
- No
  - She bumps into a tree, is knocked unconscious and wakes up 3 hours later.

Should she stay there and eat her sandwich?
- Yes
  - The honey and rhubarb sandwich is delicious. It is really getting late.
- No
  - She finds a clearing, there is a sword sticking out of a rock there.

Should she go home?
- Yes
  - It turns out to be a valuable artifact. She sells it and donates the proceeds to help bears everywhere.
- No
  - She bumps into a tree, is knocked unconscious and wakes up 3 hours later.

Figure 1: Flowchart for Fluffy Bear Story
Question 7: Write a function called *striped_rectangle* that takes four parameters as arguments: *character1*, *character2*, *width* and *height*. It should draw a striped rectangle consisting entirely of *characters* 1 and 2. The parameters *width* and *height* should determine the width and height of the rectangle, where the width is measured in characters and the height is measured in lines. The lines should alternate between being entirely of *character1* or entirely of *character2*. Thus if *height* is an odd number there will be one more *character1*-based line than there are *character2*-based lines. The following is an example of how the intended function should behave:

```python
>>> striped_rectangle('=', '*', 15, 6)
===============
***************
===============
***************
===============
***************
===============
```

```python
>>> striped_rectangle('=', '*', 15, 6)
```
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.

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.