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.
Answer all 4 questions in Section 1. Answer 2 questions from Section 2.

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.

This test includes 2 separate booklets.

Booklet 1 (this booklet) includes:

1. Section 1 questions with boxes to fill in your answers. Extra boxes额外 space is provided to allow room for cross-outs and corrections. You can use any part of the test (other than the boxes) for scrap paper. However, please note that the backs of pages will not be scanned into Gradescope and therefore not considered for credit so do not include answers on the backs of pages. Please do not remove any pages from the test and please do not add any pages. Extra scrap paper is available if you need it.

2. Additional pages for answering Section 2 questions. Each part 2 question is allocated 3 blank pages, labeled with the question that they are intended for. Please make an effort to use these pages to answer the corresponding questions. You can use the back of pages as scrap paper, but the backs of pages will not be scanned in to Gradescope. If for some reason, you need to write an answer in a non-obvious place, please indicate this with a clear note, somewhere in the question, e.g., see pages marked for question 7 could be a note indicating that the rest of an answer (for question 5 or 6) is found in the space allocated for question 7.

It is important that you do not remove any pages from this booklet. Removing pages may prevent Gradescope from working properly.

Booklet 2 (the other booklet) contains:

1. Section 2 questions

2. A GLOSSARY OF TERMS – 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: What is the value of the global variable output after the following code is executed.

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

Answer: '11' Answer:

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 resulted in a small deduction, but answering 2, would have resulted in an incorrect answer.
Question 1: What is the value of the global variable output, after the code above is executed? Put your answer in the box provided. If you make an error, cross it out clearly and write the correct answer in the second box.

```python
def spooner(words):
    word_list = words.split(' ')
    if len(word_list) < 2:
        print('Invalid list for Spoonerisms.')
    else:
        first_word = word_list[0]
        last_word = word_list[-1]
        vowel1 = 'not yet'
        vowel2 = 'not yet'
        index1 = 0
        index2 = 0
        while (vowel1 == 'not yet') and (index1 < len(first_word)):
            if first_word[index1] in 'aeiou':
                vowel1 = index1
                index1 = index1 + 1
        while (vowel2 == 'not yet') and (index2 < len(last_word)):
            if last_word[index2] in 'aeiou':
                vowel2 = index2
                index2 = index2 + 1
        if (vowel1 != 'not yet') and (vowel2 != 'not yet'):
            new_first_word = last_word[:vowel2]+first_word[vowel1:]
            new_last_word = first_word[:vowel1]+last_word[vowel2:]
            output = new_first_word+
            for word in word_list[1:-1]:
                output = output+' '+word
            output = output+' '+new_last_word
            return(output)

output = spooner('lack of pies')
```

Answer: 

Answer: 

Answer: 

Answer:
Question 2 What prints out when the following code is executed? Indicate this in the boxes provided.

```python
def mirror(instring):
    outstring = ''
    for num in range(len(instring)):
        outstring = outstring + instring[(-1*num)-1]
    return outstring

def mirror_mirror(inlist):
    outlist = []
    for word in inlist:
        outlist.append(mirror(word))
    outlist.reverse()
    return(outlist)

def mirror_statement(instring):
    input_list = instring.split(' ')
    out_list = mirror_mirror(input_list)
    output = ''
    for word in out_list:
        output = output+word+''
    output = output[:-1]
    return(output)

def main():
    for input_string in ['mood sees doom','stop desserts pots','Bob knits live lager']:
        output = mirror_statement(input_string)
        print('in:',input_string)
        print('out:',output)
main()
```

Answer: 3
Question 3
Question: In first answer box on the next page, draw (approximately) the configuration that would be drawn by the following turtle program. If you make an error, cross out the incorrect answer and redraw the answer in the second box.

```python
import turtle

def initialize_turtle():
    global my_screen
    global my_turtle
    my_screen = turtle.Screen()
    my_turtle = turtle.Turtle()

def dashed_line(distance, increment):
    distance_left = distance
    up = False
    while distance_left > 0:
        if up:
            my_turtle.pd()
        else:
            my_turtle.pu()
        up = not(up)
        movement = min(increment, distance_left)
        my_turtle.fd(movement)
        distance_left = distance_left - movement

def dashed_x(distance, increment, angle):
    if angle > 89:
        angle = 89
    elif angle < 1:
        angle = 1
    my_turtle.pu()
    my_turtle.setposition(distance/2, 0)
    my_turtle.left(angle)
    dashed_line(distance, increment)
    my_turtle.pu()
    currentx, currenty = my_turtle.pos()
    my_turtle.setposition(currentx, 0)
    my_turtle.left(180 - (2 * angle))
    dashed_line(distance, increment)

def main():
    initialize_turtle()
    dashed_x(150, 10, 45)

main()
```
Question 4: What prints out (include the result of all `print` statements) when the following code is executed? Put the answers in the boxes below. As before, if you make a mistake in the first answer box, cross it out and put your answer in the second box.

```python
stuff_to_share = [[1,’banana’],[66,’duck’],[48,’boat’],[100,’clamato’],
[2,’pencil’],[57,’t-shirt’],[90,’antique thing’]]

def sharing_program(stuff,people):
    import os
    stuff.sort()
    stuff.reverse()
    person_number = 0
    totals = []
    for num in range(people):
        totals.append([0,num+1])
    for item in stuff:
        print(person_number+1,’gets the’,item)
        totals[0][0] = totals[0][0]+item[0]
    totals.sort()
    for total_item in totals:
        print(’Person’,total_item[1],’got’,total_item[0],’worth of stuff.’)

sharing_program(stuff_to_share,2)
```

Answer: 6
Answer:
Section 2 (50 points): Answer 2 of the 3 questions in this section (see booklet 2 for the questions). For each question, you do, write a Python program as specified. Please include the code in the pages allocated to that question, e.g., answer question 7 on the pages marked Question 7. Cross out any code that you do not want counted. If you do 3 (rather than 2) questions, cross out the answer to the question that you do not want to be counted. If for some reason, you need to write an answer in a non-obvious place, please indicate this with a clear note, somewhere in the question, e.g., see pages marked for question 7 could be a note indicating that the rest of an answer (for question 5 or 6) is found in the space allocated for question 7.

Question 5: Page 1