Note: This sample test would take substantially more time to complete than the actual midterm will. There are a total of 12 questions in this sample, including 3 section 3 questions (which are the most time-consuming). There will be at most 9 questions on the actual midterm. In particular, there will be no more than 5 section 1 questions; no more than 2 section 2 questions; and no more than 2 section 3 questions. The goal of this sample is to provide a good sampling of the types of questions that might be on the test and the range of topics. Individual questions may be harder or easier than the ones on the actual midterm (question difficulty is hard to define precisely.). Note that sections 1 and 3 contain questions of varying levels of difficulty.

There are 3 sections, each worth 35 points. The maximum score on the test will be 105.

It is essential that you **PUT YOUR NAME 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.

**Section 1**: Below you will find several Pieces of Code followed by a question and a place to fill in an answer. Assume that there are no bugs in the code that will make the system crash, although some code may not solve a particular 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 does output equal?
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 print_5s():
    print(5,5,5, sep='*', end='**')
    print(5,5,5)

print_5s()
```

Question: What prints out by executing the code above (approximately)?
Answer:

**Question 2**

```python
def nine_complement(digit):
    return(9 - digit)

def nine_complement_string(number_string):
    output = ''
    for digit in number_string:
        output = output+str(nine_complement(int(digit)))
    return(output)

output = nine_complement_string('0123456789')
```

Question: What does output equal?
Answer:
Question 3

```python
output = (float('5'+0) * 7)
```

Question: What does output equal after the code above executes?
Answer: 

Question 4

```python
output = 50
for number in range(6):
    output = output - number
```

Question: What does output equal after the loop executes?
Answer: 

Question 5

```python
def diagonal_asterisks(length):
    for spaces in range(length):
        print(' ' * spaces, '*', sep='')

diagonal_asterisks(7)
```

Question: What would the above function call print out?
Answer: 

Question 6

def vending_machine(money, price):
    if (money < price):
        print('Not enough money')
    elif (money == price):
        print('There is no change.')
    else:
        dollars = 0
        quarters = 0
        dimes = 0
        nickels = 0
        pennies = 0
        change = money - price
        dollars = change // 1
        change = change - dollars
        change = change * 100  ## converting to cents
        ## intial amount of money is in dollars
        ## for coins smaller than a dollar,
        ## is is more convenient to calculate
        ## in terms of cents.
        if change > 0:
            quarters = change // 25
            change = change - (quarters * 25)
            if change > 0:
                dimes = change // 10
                change = change - (dimes * 10)
                if change > 0:
                    nickels = change // 5
                    change = change - (nickels * 5)
                    pennies = change
        print('Your change is: ')
        if dollars > 0:
            print(dollars, 'dollars')
        if quarters > 0:
            print(quarters, 'quarters')
        if dimes > 0:
            print(dimes, 'dimes')
        if nickels > 0:
            print(nickels, 'nickels')
        if pennies > 0:
            print(pennies, 'pennies')

output = vending_machine(5.00, 4.34)

Question 6a: What would be printed by the print statements produced by the code above?
Answer:

Question 6b: What would does the global variable output equal after the code above executes?
Answer:
Question 7:

import turtle
my_screen = turtle.Screen()
turtle1 = turtle.Turtle()
turtle2 = turtle.Turtle()
turtle1.speed(0)
turtle2.speed(0)

def move_both_turtles(distance):
    turtle1.fd(distance)
turtle2.fd(distance)

def turn_turtles_opposite(degrees):
    turtle1.left(degrees)
turtle2.left(degrees+180)

def square(turtle, side):
    for rep in range(4):
        turtle.fd(side)
turtle.left(90)

def turtle_square_eight():
    turtle1.pd()
turtle2.pd()
    turn_turtles_opposite(45)
square(turtle1, 40)
square(turtle2, 40)
turtle1.pu()
turtle2.pu()

turtle_square_eight()
Section 2: This section consists of 2 sample pieces of code containing syntax errors. On the appropriate line numbers in the table below the function, please list the errors (if any), along with possible corrections. Sample output is provided to indicate what the program is intended to do.

A minimum number of errors is listed. You do not need to find more errors than that number. For example, if the description says to find three errors, you only need to find three, even if there are actually four or five errors. If you identify more errors than the minimum, you will be graded on the total number of errors that you list. For example if you list 3 actual errors and you are mistaken about a fourth one, you will get 3/4 of the total number of points available.

The comments are meant to be instructive. Please do not look in the comments for errors. Assume that the CONTENTS of comments are error-free (but of course syntactically incorrect comments should be marked as incorrect).

Sample Question B: The function minus takes two numbers as input and subtracts the second from the first. **There is only one error.**

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td><code>def minus(total, decrement)</code></td>
</tr>
<tr>
<td>2.</td>
<td><code>output = total - decrement</code></td>
</tr>
<tr>
<td>3.</td>
<td><code>return(output)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Line</th>
<th>Correction of Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>need to add colon (:) at end of line</td>
</tr>
<tr>
<td>2.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
</tr>
</tbody>
</table>

Question 8: The input to this function is a string (Things). Things is inserted into two sentences that involve complaints. Each of these sentences are then printed out. **Find 3 errors.**

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td><code>def complain_about(Things):</code></td>
</tr>
<tr>
<td>2.</td>
<td><code>output_string1 = 'I really don't appreciate ' + Things</code></td>
</tr>
<tr>
<td>3.</td>
<td><code>output_string2 = 'I think ' + Things + ' are unfair'</code></td>
</tr>
<tr>
<td>4.</td>
<td><code>print(output_string)</code></td>
</tr>
<tr>
<td>5.</td>
<td><code>print(output_string2)</code></td>
</tr>
<tr>
<td>6.</td>
<td><code>## Example with Things set to: tests</code></td>
</tr>
<tr>
<td>7.</td>
<td><code>## I really don't appreciate tests</code></td>
</tr>
<tr>
<td>8.</td>
<td><code>## I think tests are unfair.</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Line</th>
<th>Correction of Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td></td>
</tr>
</tbody>
</table>
Question 9: The length of a person’s foot is claimed to be 15% of their height. This function tests this hypothesis for one individual. It prints out an informative statement and returns the user’s foot to height ratio. **Find 4 errors**

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>def test_foot_to_height_ratio():</td>
</tr>
<tr>
<td>2.</td>
<td>print('This function calculates and records your foot-length to total height ratio.')</td>
</tr>
<tr>
<td>3.</td>
<td>print('All measurements should be given in inches if you are American</td>
</tr>
<tr>
<td></td>
<td>or in millimeters if you are from anywhere else in the world.')</td>
</tr>
<tr>
<td>4.</td>
<td>foot-length=float(input('Please measure your foot and state its length: '))</td>
</tr>
<tr>
<td>5.</td>
<td>height=input(float('Please indicate your total height: '))</td>
</tr>
<tr>
<td>6.</td>
<td>ratio = foot-length / height</td>
</tr>
<tr>
<td>7.</td>
<td>print('Thank you for your participation in our study!!!!')</td>
</tr>
<tr>
<td>8.</td>
<td>print('Your foot-length to height ratio is: ', ratio)</td>
</tr>
<tr>
<td>9.</td>
<td>print('The average foot-length to height ratio may be about .15')</td>
</tr>
<tr>
<td>10.</td>
<td>print('Your ratio is ', (ratio / .15) 'times that ratio')</td>
</tr>
<tr>
<td>11.</td>
<td>return(ratio)</td>
</tr>
<tr>
<td>12.</td>
<td>## Example print statements given user input of 11 and 72:</td>
</tr>
<tr>
<td>13.</td>
<td>## Thank you for your participation in our study!!!!</td>
</tr>
<tr>
<td>14.</td>
<td>## Your foot-length to height ratio is: 0.1527777777777778</td>
</tr>
<tr>
<td>15.</td>
<td>## The average foot-length to height ratio may be about .15</td>
</tr>
<tr>
<td>16.</td>
<td>## Your ratio is 1.0185185185185186 times that ratio</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Line</th>
<th>Correction of Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td></td>
</tr>
<tr>
<td>16.</td>
<td></td>
</tr>
</tbody>
</table>
Section 3: Write Programs as specified.

Question 10: This program calculates the amount of fabric needed to create a sweater for a dog. For simplicity, the program calculates the amount of material required to make a hollow cylinder, closed at the top and bottom, based on some measurements of the dog. Of course when the sweater is made, holes will be cut into the cylinder for the dog’s head and legs, and other adjustments will be made so it fits properly.

The height of the cylinder is the distance from the base of the neck to the base of the tail. The diameter is the height of the thickest portion of the dog’s chest. The formula for the area of the surface of a cylinder is \( (height \times \pi \times diameter) + (2 \times \pi \times radius^2) \). The program should ask the user for the appropriate input and print the appropriate output in a way the user can understand. The function should return the total amount of fabric in square inches or square centimeters.
**Question 11**: This program will center three lines of text as follows. The user is asked to provide three sentences of no more than 65 characters. The program then centers each line by adding between 0 and 32 spaces to the beginning and then printing it. For example, if one of the sentences was ‘Hello World’, the system would add 27 spaces to the beginning of the string before printing. Note that the function: `len(string)` returns the number of characters in a string. Furthermore, it may be necessary to consistently round the number of spaces added either up or down in order to center the strings consistently. Thus the centering will be as close as possible, but not perfect.
**Question 12:** Write a program that will ask a user several questions about how nice they are. The program will then calculate how nice they are in Standard Niceness Units (SNUs), according to guidelines devised by the World Council on Niceness (WCN). These guidelines are represented in the flowchart below. The flowchart indicates when questions should be asked and how many SNUs are added or subtracted to the total SNU score based on the answers. The questions are given as global variables. You should use these variables instead of writing out all the questions (it will save a lot of time). At your option, you may also use (without writing it yourself), the function `yes_or_no`. You can vary from the flowchart if it doesn’t matter logically. For example, you could ask the user questions 1, 2, 3, 4, 5 and 6 and use the answers (stored as variables) in the if/else statements of your decision tree if this seems easier to you.

The `yes_or_no` function and global variables:

```python
def yes_or_no(question):
    answer = ''
    while not (answer in ['Yes','yes','No','no']):
        print('Please answer the following question "yes" or "no". ')
        answer = input(question)
        if answer in ['Yes','yes']:
            return(True)
        else:
            return(False)
```

```python
question1 = 'Do you always share your candy? '  
question2 = 'Do you clean up after yourself most of the time? '  
question3 = 'Do you give money to charity? '  
question4 = 'Do you spend some of your time doing charitable work? '  
question5 = 'Do you care about non-human animals? '  
question6 = 'Would you commit a crime against another person for personal gain? '  
question1a = 'Do you always share your food? '  
question2a = 'Do you clean up after other people? '  
question3a = 'Would you give up all your money to help a friend in serious need? '  
question3a1 = 'Would you do the same for a stranger? '  
question4a = 'Would you dedicate your life to a charitable cause and be poor yourself? '  
question5a = 'Do you believe in eating all non-human animals? '  
question5a1 = 'Do you believe in eating some non-human animals? '  
question6a = 'Would you steal so much that your victim becomes poor? '  
question6b = 'Would you murder another person for profit? '
```

Some hints for following the flow chart: questions 1,2,3,4,5,6 are mandatory. Regardless of whether question 1 is answered with 'yes' or 'no', question 2 must be asked. Regardless of whether question 2 is answered 'yes' or 'no', question 3 must be asked, etc. However, other questions are only asked under certain circumstances. In most cases a positive answer yields more questions, but not always. Usually, positive answers cause points to be added to the SNU (total Standard Niceness Units), but occasionally, positive answers cause points to be subtracted.

The flow chart is on the following page.

**For extra credit**, you can complicate the decision tree further by adding five additional questions. Only add questions that would be asked if previous questions are answered No.
Initialize SNU to 0

Start

Ask Question 1

Yes

Ask Question 1a

Add 10 to SNU

Ask Question 1a

No

Question 1

No

Add 10 to SNU

Ask Question 2

Yes

Add 10 to SNU

Ask Question 2a

No

Question 2

No

Add 10 to SNU

Ask Question 3

Yes

Add 10 to SNU

Ask Question 3a

No

Question 3

No

Add 10 to SNU

Ask Question 4

Yes

Add 10 to SNU

Ask Question 4a

No

Question 4

No

Add 10 to SNU

Ask Question 5

Yes

Add 10 to SNU

Ask Question 5a

No

Question 5

No

Add 10 to SNU

Ask Question 6

Yes

Subtract 30 from SNU

Ask Question 6a

No

Ask Question 6b

Yes

Subtract 30 from SNU

Ask Question 6a

No

Question 6

No

Subtract 50 from SNU

Ask Question 6b

Yes

Subtract 50 from SNU

Ask Question 6b

No

Question 6

No

Subtract 3 from SNU

Add 10 to SNU

Print Results

Return Total Score

End