General notes:

- When writing programs, comment your code if you think it may be unclear.
- You will lose a small # of points for syntax errors so double-check your work to ensure that it is syntactically correct.
- No calculators or any other devices. Just a pencil and your mind.
- If you need more space, use the back of each page.
1. Trace the output of the following code samples (8 points)

```python
for i in range(5, 10):
    print ("num =", i, end=" ")

num = 5 num = 6 num = 7 num = 8 num = 9
```

```python
for i in [5, 10]:
    print ("num =", i, end=" ")

num = 5 num = 10
```

```python
for i in range(5, 10, 3):
    print ("num =", i, end=" ")

num = 5 num = 8
```

```python
for i in range(5, 10, 3):
    for j in range(2, 4):
        print ("num 1 =", i, "- num 2 =", j)

num 1 = 5 - num 2 = 2
num 1 = 5 - num 2 = 3
num 1 = 8 - num 2 = 2
num 1 = 8 - num 2 = 3
```
2. Given the following Strings:

    good = "This midterm rocks!"
    bad = "This midterm sucks?"

Trace the output (10 points):

    print (good[0])
    T

    print (good[4])
    -

    print (bad[:4])
    This

    print (good[len(bad)-3])
    k

    print (good.isalpha())
    False

    print (bad[-5:].upper())
    UCKS?

    print (bad[0].isalpha())
    True

    print (bad[-1:].isalpha())
    False

    print (good.find(bad[-4:-2]))
    15

    print (bad[5:8] in good)
    True
3. Trace the output of the following programs (6 points):

```python
def signJam(x):
    print("**+*" * x)

password = "i4an1"

for i in password:
    if i.isdigit():
        signJam(int(i))

*+**++**+***
*+*

def fiddle(x):
    print("1":, x)
    x -= 3
    sticks(x)
    print("1":, x)

def sticks(x):
    print("2":, x)
    x -= 1
    print("2":, x)
    return x

x = 3
fiddle(x + 2)
print("3":, x)
```

```
1: 5
2: 2
2: 1
1: 2
3: 3
```
Questions #4 through 8 are programming questions. Select 4 of these questions to solve (6 points each)

4. Business is booming at ‘House of Cards’, a local business card shop, and they want you to make them a function to automate their business card online ordering system. There are basic rates for cards with extras that users can purchase.

- Business cards cost $10.00 for the first 100 cards.
- Every additional 100 cards are $5.00.
- Black ink is considered ‘1-color print’ and that’s included
- Every color thereafter is $2.00 per 100 cards
- If they want glossy paper, it’s $5.00 per 100 cards

Write a FUNCTION called card_calc that accepts 3 ARGUMENTS – the number of cards, the amount of colors, and glossy paper or not. Compute the total cost of an order using these three inputs and RETURN it. Note that if any input is invalid (i.e. # of cards isn’t a multiple of 100, less than one color of ink) then your function should return the string “INVALID ORDER”. Here’s a sample program to show your function in action.

print ( card_calc(300, 3, “y”) ) >> 47.0
print ( card_calc(300, 1, “bazooka”) ) >> INVALID ORDER
print ( card_calc(600, 5, “n”) ) >> 83.0

def card_calc(amount, colors, gloss):
    if amount % 100 == 0:
        amountCost = amount/100 * 5 + 5.00
    else:
        return "INVALID ORDER"

    if colors == 1:
        colorCost = 0
    elif colors > 1:
        colorsCost = amount/100 * ((colors - 1) * 2)
    else:
        return "INVALID ORDER"

    if gloss == "y":
        glossCost = amount/100 * 5.00
    elif gloss == "n":
        glossCost = 0
    else:
        return "INVALID ORDER"

    return amountCost + colorsCost + glossCost
5. Write a program that continually prompts the user to enter in specifications for a business card order (amount, number of colors, and glossy or not). Then use the “card_calc” function written for the previous question to compute the cost of the desired order. If the order is valid you should add the cost of the order to the user’s bill and provide them with a running total of their bill. If the order isn’t valid you should report this. Allow the user to continue to enter in orders until they choose to end the program. Here’s a sample running of your program:

Your current bill is $0
How many business cards would you like? 300
How many colors should the cards be? 3
Do you want glossy cards? Type 'y' or 'n': y
This order will cost $47.0
Type 'yes' to place another order: yes

Your current bill is $47.0
How many business cards would you like? 300
How many colors should the cards be? 3
Do you want glossy cards? Type 'y' or 'n': bazooka
This is not a valid order
Type 'yes' to place another order: no

Your total bill is $47.0

order = "yes"
bill = 0

while order == "yes":
    print("Your current bill is $", bill, sep="")
    cards = int(input("How many business cards would you like? "))

    colors = int(input("How many colors should the cards be? "))

gloss = input("Do you want glossy cards? Type 'y' or 'n': ")

totalCost = card_calc(cards, colors, gloss)

    if totalCost == "INVALID ORDER":
        print("That's not a valid order."")
    else:
        print("This order will cost $", totalCost, sep="")
        bill += totalCost

    order = input("Type 'yes' to place another order: ")
    print()

print("Your total bill is $", bill, sep="")
6. A cipher is a coded way of writing. You’ve been tasked with writing a simple but effective cipher to hide text that’s sent. You will create a **FUNCTION** that accepts one **ARGUMENT**: an input string. It will **RETURN** the encoded message. When encoding, the function takes an input string and encodes each character into its ASCII value * 2, and returns each of those values with a space between each. HINT: when you turn character to ASCII value, the numbers are integer values, not strings, so keep in mind when concatenating everything together to return the value.

Besides the function itself, prompt the user for input string, execute the function, and print the return value like below. Don’t worry about making sure the inputted data is accurate or making it prompt continuously.

Sample input/output:
What do you want to encode: **You’ll never catch me!**

You encoded message is: 178 222 234 78 216 216 64 220 202 236 202 228 64 198 194 232 198 208 64 218 202 66

def cipher(phrase):
    newPhrase = ""
    for i in phrase:
        newPhrase += str(ord(i)*2) + " 
    return newPhrase

phrase = input("What do you want to encode: ")
newPhrase = cipher(phrase)
print(newPhrase)
7. Kristof Vandelfuss is a great writer. But he hates everything that isn’t glorious letters of the alphabet. As you know, he’s getting older, a little crazier, and his family has asked you to write a program that strips all pieces of text of everything that isn’t letters of the alphabet and spaces (he’s cool with space). Therefore, he should never see any numbers, or any punctuation, etc. The program should consist of a FUNCTION that takes a single input ARGUMENT: a string. It should RETURN two values: the letters of the alphabets/spaces for him to see, and, for his family, the numbers and punctuation that was subtracted from it.

Besides the function itself, prompt the user for input string, execute the function, and print the return value like below. Don’t worry about making sure the inputted data is accurate or making it prompt continuously.

Sample input/output:

Enter the text to run through the Vandelfusser: _____

Kristof’s text is: It Is gr Be near this crazydude
But don’t show him: ..!8.2.,;,…!

def wordSplice(phrase):
    newPhrase = "
    secret = "
    for i in phrase:
        if i.isspace() or i.isalpha():
            newPhrase += i
        else:
            secret += i
    return newPhrase, secret

phrase = input("Enter the text to run through the Vandelfusser: ")
print()

new, hidden = wordSplice(phrase)
print("Kristof’s text is: ", new)
print("But don’t show him: ", hidden)
8. John Smith is a pretty insecure dude because his name is a little funny, and he gets very overshadowed by John Smith. He wants to pay you to write a program that will sort through long pieces of text and identify if John or Smith is in it. If either are, he wants to know the index number of those words in the string, then he wants the words replaced with his name (don’t worry about multiple occurrences of either name). This doesn’t need to be a function, but should keep re-prompting.

Sample input/output:

Enter the text to run through the Simther: I once met John, he was such a brat, but not a Smith.

John appears at index: 11
Smith appears at index: 47
New text: I once met Jonh, he was such a brat, but not a Simth.
Type ‘yes’ to test more text: yes
Enter the text to run through the Simther: Sea shells by the sea shore.

There is no John or Smith in this text.
Type ‘yes’ to test more text: yes
Enter the text to run through the Simther: I once knew a man named John.

John appears at index: 24
New text: I once knew a man name

Type ‘yes’ to test more text: no

goAgain = 'yes'

while goAgain == 'yes':
    text = input("Enter the text to run through the Simther: ")
    print()
    if "John" not in text and "Smith" not in text:
        print("There is no John or Smith in this text.")
        goAgain= input("Type ‘yes’ to test more text.")
        continue

    if "John" in text:
        johnPos = text.find("John")
        print("John appears at:", johnPos)
        text = text.replace("John", "Jonh")

    if "Smith" in text:
        smithPos = text.find("Smith")
        print("Smith appears at:", smithPos)
        text = text.replace("Smith", "Simth")

    print("New text:", text)
    goAgain = input("Type ‘yes’ to test more text: ")