1. What will this print?

```python
def foo(n):
    for i in range(0, n):
        print("\*", end="")
        print("!")

def bar(n):
    for i in range(0, n):
        foo(2*i)

def main():
    bar(3)

main()
```

Answer:

```
!
**!
****!
```
2. The following program is supposed to decide whether numbers between 5 and 10 are prime. It prints

5 True
6 False
7 True
8 False
9 True

Oops; 9 is not actually prime.

a) Explain why this program says 9 is prime.
b) In general, for which numbers n will isPrime(n) be True?

def isPrime(n):
    for d in range(2, n):
        if n%d == 0:
            return False
        else:
            return True

for x in range(5, 10):
    print(x, isPrime(x))

Answers:

a) In isPrime(9), the for-loop starts with d at 2>
   Since 9%2 is 1, the function returns True AND WE LEAVE THE FUNCTION. So it prints 9 True
b) isPrime( ) will return True for all odd numbers.
3. How can we fix the following program? It is supposed to let the user enter positive numbers and then print the average of those numbers. When I give it numbers 1, 2, 3, and then -1 to exit it reports the average as 1.67. I am pretty sure the average of 1, 2, and 3 is 2.0. Change the program so it correctly calculates the average.

```python
sum = 0
count = 0
done = False
while not done:
    x = eval(input("Enter a number, or -1 to exit: "))
    sum = sum + x
    if x < 0:
        done = True
    else:
        count = count + 1
print( "The average is %.2f"%(sum/count))
```

Answer: Move the `sum = sum + x` line to inside the else statement, either before or after `count = count + 1`
4. What will the following program print?

```python
def main():
    print( f(3) )

def f(x):
    for y in range(0, x):
        return g(y)

def g(x):
    if x > 0:
        return x-1

main()
```

Answer: This is tricky. `main()` prints `f(3)`. Function `f()` has a for-loop, but the first thing it does is return `g(0)` and since it returns, we leave `f()` after that. So `f(3)` returns `g(0)`. The problem is that function `g()` doesn’t say what to return when `x` is 0, so it returns `None`. Altogether, the program will print `None`. 
5. The following function gets an error message on the line
   for y in range(Start, x)
The message says that variable Start is undefined. Explain this message. Doesn’t the line
   Start = 3
define Start? Note that I am not asking you to change the program; just explain this error
message.

def printer(x):
    # This prints the numbers from Start up to x
    for y in range(Start, x):
        print( y, end=" ")
    print()

def main():
    Start = 3
    printer(5)

main()

Answer: One function can’t see another function’s variables. The Start variable
in main( ) is a completely different variable than the Start variable in printer( ).
So No, the line Start=3 in main( ) doesn’t define Start in printer( ). Since nothing
in printer( ) gives value to Start, it is undefined.
6. Here is a sequence of numbers: 0, 1, 3, 6, 10, 15, 21, 28, ...

Note that the first pair (0 and 1) differ by 1, the second pair (1 and 3) differ by 2, the next pair (3 and 6) differ by 3, and so forth. Write a program that asks the user for a number \( n \) and then prints the first \( n \) elements of this sequence. You can print them horizontally or vertically; I don’t care about the format as long as you print the correct numbers.

Answer: I did this slowly, where I could see what everything means. In my solution \( x \) is the variable we print, \( \text{diff} \) is the difference between one element of the sequence and the next:

```python
n = eval(input("Number: "))
x = 0
diff = 1
for i in range(0, n):
    print(x)
    x = x + diff
    diff = diff+1
```

A lot of people in the class did it this way:

```python
n = eval(input("Number: "))
total = 0
for i in range(0, n):
    total = total + i
print(total)
```
7. Write function `duplicates(s)` that returns the number of duplicate letters in string `s`. For example, `duplicates(“abacaba”)` is 4 because the initial ‘a’ is repeated 3 times after its initial appearance and the ‘b’ is repeated once. Similarly `duplicates( “bob” )` is 1 and `duplicates( “abba” )` is 2.

Here is one way: a letter is a duplicate if it appears later in the string:

```python
def duplicate(s):
    count = 0
    for i in range(0, len(s)):
        if s[i] in s[i+1:]:
            count = count + 1
    return count
```

Here’s another version that doesn’t use the “in” operator. It uses a break statement to break out of the inner loop as soon as it finds a match; without that it will overcount. You could also do the inner loop as a while-loop instead of using a break statement.

```python
def duplicate(s):
    count = 0
    for i in range(0, len(s)):
        for j in range(i+1, len(s)):
            if s[i] == s[j]:
                count = count + 1
                break
    return count
```
You can use this space for extra work on any problem. If you want me to grade anything here, indicate clearly which problem you are referring to.
Please write and sign the Honor Pledge when you have finished the exam.