• This little program:

```python
bmi = raw_input("Enter body mass index: ")
inches = 70 # average height is 70 inches
print "I guess your weight is: 
print (bmi*inches*inches)/703
```

a) correctly prints out a guess for the user’s weight.
b) might crash, depending on what the user enters.
c) always crashes.
Answer: c. The function `raw_input()` always produces a string, and the program fails to convert the string to an integer before doing arithmetic with it.

• This program:

```python
rate = 13.0
weeklyRate = rate/52
print weeklyRate
```

a) prints 0.25
b) prints 0
c) prints 13.0
Answer: c. The variable rate does not get changed by this program. The variable weeklyRate will contain 0.25.
This question appeared only on Test form A.

• This program:

```python
rate = 13
weeklyRate = rate/52
print weeklyRate
```

a) prints 0.25
b) prints 0
c) prints 13.0
Answer: b. Because 13 is an integer, the result is the integer part of the result (52 goes into 13 zero times).
This question appeared only on Test form B.

• The following one-line program:

```python
print "Cozy"+"Shack"
```

a) prints Cozy Shack
b) prints CozyShack
c) prints Cozy+Shack
c) will crash.
Answer: b. The "+" here is the string concatenation operator.

• After running this program:

```python
debt = 1000.0
balance = debt
balance = balance + (16.0/100.0)*balance
```

a) debt contains 1000.0.
b) debt contains 1160.0.
c) debt contains balance.
Answer: a. Again, the variable debt is not changed by this program. The variable balance contains 1160.0.

• This program:

```python
x = raw_input("Enter a or b: ")
if x == "b":
    points = 1
else:
    points = 0
print "Points: ", points
```

a) always prints out the score.
b) might print out the score, depending on what the user types.
c) might crash, depending on what the user types.
Answer: b. The score will only get printed if the program executes the block under the "else".
• This program:

```python
import random as random
x = random.randrange(0,4)
if ((x > 2) and (x < 1)):
    print x
```

a) might print 1 or 2
b) might print 0 or 3
c) will never print anything

Answer: c. The variable x cannot be both >2 and <1.

• The following program should count from 0 to 8 by 2's, that is, it should print out 0,2,4,6,8. Which Boolean condition could fill in the blank in the while statement?

```python
count = 0
increment = 2
while ________________ :
    print count
    count = count+increment
```

a) count < 5
b) count < 8
c) count < 10
d) none of the above.

Answer: c. The variable count should be printed when count = 8.

• The following program should ask for input until the user types either Y or N. Which Boolean condition could fill in the blank in the if statement?

```python
acceptableInput = False
while not acceptableInput :
    reply = raw_input("Enter Y or N: ")
    if _____________:
        acceptableInput = True
```

a) (reply == "Y") or (reply == "N")
b) (reply == "Y") and (reply == "N")
c) acceptableInput
d) not acceptableInput

Answer: a. We want to assign acceptableInput = True when either one of the acceptable answers is given.

This question only appeared on Test Form A.

• After running the following program:

```python
goodInput = False
while not goodInput:
    reply = raw_input("Enter H or T: ")
    if (reply == "H") or (reply == "T"): 
        goodInput=True
print "Thank you."
```

a) the variable goodInput will contain True.
b) the variable goodInput will contain False.
c) the variable goodInput may contain either True or False.

Answer: a. The program only ends when the user enters H or T; at which point the variable goodInput will get assigned the value True. This question only appeared on Test Form B.

• The following program should print Score is 10. (notice that there is no space before the period). What could fill in the blank?

```python
s = 10
print ________________
```

a) "Score is",s,"."
b) "Score is",str(s),"."
c) "Score is",s+"."
d) "Score is",str(s)+"."

Answer: d. Choices a,b run but leave a space inbetween the number and the period. Choice c crashes because s is an integer and "." is a string.

• The following program will print:

```python
age = 19
duis = 2
if (age<21) or (duis>0):
    print "High-risk"
else:
    print "Low-risk"
```

a) High-risk
b) Low-risk
c) nothing.
d) both High-risk and Low-risk.

Answer: a. The or operator produces True when the Booleans being or’d together are both True, or when one or the other of them is True.
• This program...

```python
balance = 1000.0
payment = 200.0
while balance > 0:
    print "Balance is",balance - payment
```

a) prints five lines.
b) prints four lines.
c) enters an infinite loop.
d) crashes.

Answer: c. Since the value in variable `balance` is never changed, the loop runs forever.

• This program...

```python
name = raw_input("Enter your name: ")
if (name == "):  
    raw_input("Press enter to exit.")
print "Hello," , name 
raw_input("Press enter to exit.")
```

a) always prints Hello,
b) may or may not print Hello, depending on what the user enters.
c) never prints Hello,
d) may crash, depending on what the user enters.

Answer: a. The statement that prints Hello, is not in the block under the if statement. Also, the statement `raw_input("Press enter to exit.")` does not exit the program. It just waits for user input (and then throws that input away).

• This program prints...

```python
size = 7
factor = 7/2
while size > 0:
    print size
    size = size - factor
```

a) two lines
b) three lines
c) four lines

Answer: b. The program prints 7,4,1.
Programming problem.

We observed in class that an investment at 7%, compounded monthly, more than doubles its value after ten years. In finance, they would say the "doubling time" for 7% compounded monthly is ten years. Write a short program that takes an annual interest rate as input, and figures out the doubling time if it is compounded monthly. Here is an example of what the output of your program should look like (for a very large interest rate of 60%, just to keep the output short):

```
Enter annual interest rate: 60
The monthly interest rate is 5.0 percent
Month= 1 Balance= 105.0
Month= 2 Balance= 110.25
Month= 3 Balance= 115.7625
Month= 4 Balance= 121.550625
Month= 5 Balance= 127.62815625
Month= 6 Balance= 134.009564062
Month= 7 Balance= 140.710042266
Month= 8 Balance= 147.745544379
Month= 9 Balance= 155.132821598
Month= 10 Balance= 162.889462678
Month= 11 Balance= 171.033935812
Month= 12 Balance= 179.585632602
Month= 13 Balance= 188.564914232
Month= 14 Balance= 197.993159944
Month= 15 Balance= 207.892817941
The doubling time is 1 years and 3 months
```

Some details: There are a number of ways to do this using a `while` loop. You should print out the doubling time at the end in years and months, just like in the example. Notice that the monthly interest rate is a floating-point number.

We will not take off points if entering something that is not a number causes an error, or if some crazy user input causes the program to enter an infinite loop. Comments are required, and may help us figure out what you are trying to do and so raise your score.
As usual, there are lots of correct ways to write this program. Here is one.

```python
# Get user input
answer = raw_input("Enter annual interest rate: ")
# Do not bother to check if input can be converted. This would not be
# OK in a real program.
annualRate = float(answer)
monthlyRate = annualRate/12.0
print "Monthly rate is",monthlyRate
# Put values into the other variables well need in the while loop.
principal = 100.00
balance = principal
month = 0
while balance < 2*principal:
    month = month+1
    balance = balance + balance*monthlyRate/100
    print "Month =",month,"Balance =",balance
# Get number of years using integer division
years = month/12
# Get leftover months using the remainder operator
leftOverMonths = month%12
print "The doubling time is",years,"years and",leftOverMonths,"months."
```