For the following questions, use these variable definitions.

\[
\begin{align*}
    a &= 37 \\
    b &= 10 \\
    c &= 2.1 \\
    d &= '3'
\end{align*}
\]

What is the type and value of each of the following expressions or, if it won't compile, circle that answer

<table>
<thead>
<tr>
<th>expression</th>
<th>type</th>
<th>value</th>
<th>circle if will not compile</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (2 pts) (a \div b)</td>
<td>int</td>
<td>3</td>
<td>will not compile</td>
</tr>
<tr>
<td>2. (2 pts) (a % b)</td>
<td>int</td>
<td>7</td>
<td>will not compile</td>
</tr>
<tr>
<td>3. (2 pts) ((c * 4) / 2.1)</td>
<td>float</td>
<td>4.0</td>
<td>will not compile</td>
</tr>
<tr>
<td>4. (2 pts) float(4) - 2 ** 3</td>
<td>float</td>
<td>-4.0</td>
<td>will not compile</td>
</tr>
<tr>
<td>5. (2 pts) bin( 5 )</td>
<td>str</td>
<td>'0b101'</td>
<td>will not compile</td>
</tr>
<tr>
<td>6. (2 pts) d * int(d)</td>
<td>str</td>
<td>'333'</td>
<td>will not compile</td>
</tr>
<tr>
<td>7. (2 pts) str( c )</td>
<td>str</td>
<td>'2.1'</td>
<td>will not compile</td>
</tr>
<tr>
<td>8. (2 pts) a &lt; c &lt; b or b == 1</td>
<td>bool</td>
<td>False</td>
<td>will not compile</td>
</tr>
<tr>
<td>9. (2 pts) b + c</td>
<td>float</td>
<td>12.1</td>
<td>will not compile</td>
</tr>
<tr>
<td>10. (2 pts) b &lt; 4 or not</td>
<td></td>
<td></td>
<td>will not compile</td>
</tr>
<tr>
<td>11. (2 pts) b if a &lt; b else c</td>
<td>float</td>
<td>2.1</td>
<td>will not compile</td>
</tr>
</tbody>
</table>
| 12. (2 pts) (on the next several lines) 'hi
lo''' | str   | 'hi\nlo' | will not compile |

must have quotes of some sort

spelling counts and caps count

will be str and have newline
13. (3 pts) Given these constants, complete the function, implementing its docstring.

```python
INVALID_INPUT_PROMPT = "Sorry, try again"
UNKNOWN_ERROR = 33333333
MANAGERS_NAME = 'JOE'
SENTINEL_VALUE = -99999

def killTheProgramOnUnknownError():
    ''' Ends the program
    Should be called only when an unknown error has occurred.
    Explains that the program will end and why. '''
    print( "Ending program. A unknown error has occurred." )
    exit( UNKNOWN_ERROR )
```

A programmer has been hired by a contest giving company that has very strict and well publicized age limits for entering their contests. The have hired this programmer to write a function for that takes a user's age and returns whether or not the user is old enough to enter their contest (they have only one contest).

Note that this question is NOT about branching.
Do NOT write this function.

ONLY answer this question:

14. (10 pts) What are the pieces of information this programmer needs?
   Give the source of information for each (if no information is needed, just write NO INFO).
   Note that we are NOT asking for the implementation.
   Consider both good design as well as the fact that the contest has a very specific age limit.
   Use the terminology for your answer that is used in class and in the website.
   If a piece of information might have two sources, state both.

<table>
<thead>
<tr>
<th>INFORMATION NEEDED</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>user's age (2 pt)</td>
<td>caller (2 pt) – cannot be known (-2 pt)</td>
</tr>
<tr>
<td></td>
<td>cannot be device input (-8 pt)</td>
</tr>
<tr>
<td></td>
<td>the writer of this function cannot need to know</td>
</tr>
<tr>
<td></td>
<td>where their caller got the information passed in</td>
</tr>
<tr>
<td>minimum age (2 pt)</td>
<td>caller (2 pt) OR known (2 pt) [could be either]</td>
</tr>
<tr>
<td></td>
<td>[caller would probably pass in the global]</td>
</tr>
</tbody>
</table>

15. (3 pts) What is the name of the exception that will be raise at runtime?
   In no exception will be raised, write NO ERROR.

```python
def hmm( value, min = 100 ):
    ''' ... '''
    min = min = min
    diff += abs( min - value )
    return diff
```

**UnboundLocalError** in the RHS of rightmost assignment, min has no value
there's no problem with the chained assignment
16. (8 pts) When you want to use a function written by someone else, what must you read? (circle only one letter)

a) the def line (the first line of the function definition)  
b) the docstring  
c) the body of the function  
d) the call of the function  
e) all of these  
f) a) and b) only  
g) a), b) and c) only  
h) none of these

17. (5 pts) What is returned when this function is called? (circle only one letter)

```python
def hmm( ):
    pass
```

a) Nil  
b) None  
c) Null  
d) nothing will be returned because there is no return  
e) nothing will be returned because this won't even compile  
f) nothing will be returned because NoReturnValueError will be raised at run time

18. (10 pts)
In a game, there are 20 possible prizes. The winning user's prize has already been chosen.  
Now it's time to print a line on the screen telling the winning user which prize they won.

Which problem form is controlling the output?

If you can't answer which of the problem forms, for fewer points you may answer by stating which python statement should be used to control the screen display.

- one of many, possibly none
- or - for a maximum of 4 points:
  - if-elif-elif- ... elif

ANNOUNCED DURING THE TEST:
"not everyone wins"

19. (8 pts) Which of the calls below will NOT work? (circle the letters of ALL correct answers)

```python
def gimmeMaybe( usuallyIsTen = 10, seldomTheSameValue = 10 ):
    ''' does something with parameters '''
    # code to do something with parameters
```

a) gimmeMaybe( )  
b) gimmeMaybe( 7 )  
c) gimmeMaybe( , 7 )  
d) gimmeMaybe( 7, 7 )  
e) gimmeMaybe( 10 )  
f) gimmeMaybe( , 10 )  
g) gimmeMaybe( 10, 10 )  
h) gimmeMaybe( 7, 7, 7 )
def callThee():
    ''' ... '''
0. print( 'THEE' )
1. x = 5
2. p = callMe( x )
3. print x
4. return( p )

5. def callMe( p ):
    ''' ... '''
6. print( 'ME' )
7. p = 3
8. x = p + 2
9. return 4

For example:
This arrow shows where x refers to after line 1.

PART ONE
Draw the arrow showing where p refers after line 5.

PART TWO
Draw the arrow showing where p refers after line 7.

PART THREE
Draw the arrow showing where x refers after line 8.

Write the values that will be in RAM.

PART ONE (5 pts)
PART TWO (5 pts)
PART THREE (5 pts)

PART FOUR (8 pts)

What is the output when main is called?

def main():
    print( callThee( ) )
main()

write your output here

THEE
ME
5
4
21. (15 pts) There is a built-in function you have never used before named oblatt. Here is a SnakePit session showing the help for that function.

```
IDLE 2.7.1      >>> help( oblatt )
Help on built-in function oblatt in module __builtin__:
oblatt(...)  
oblatt(object, object, object=1) -> str
The first two parameters must be strings, third must be int
Returns the str made of the first and second parameters concatenated
and repeated the third parameter number of times
E.g.: "Hi", "Lo", 3 yields the string: "HiLoHiLoHiLo" (3 "HiLo''s)
```

Obviously you cannot define this function because it is built in.

Your job is to complete drawFigureWithNames by writing the needed parameters and python statements that do what the docstring says it will do.

Even though it's not the best, the only functions you can call oblatt and other builtin functions.

You do not need to show any testing for your function.

Do NOT write a main function (obviously) or a complete program (obviously).

ONLY complete the definition for drawFigureWithNames (don't forget the parameters!)

```python
def drawFigureWithNames( first, second ):
    ''' Given two names, draws a pattern on the screen
    made of the names jammed together in a nice way.
    Here is an example of the output when the names are "John" and "Mary"
    There are always these four lines and always each name these 8 times:
    JohnMaryJohnMaryJohnMary
    -MaryJohn-
    -JohnMary-
    MaryJohnMaryJohnMaryJohn
    The two middle lines start one before where the 1st pair ends.
    Assume both parameters are strs that are names.
    '''
    print( oblatt( first, second, 3 ) )
    leftFill = ' '*(len(first)+len(second)-1) + '-'
    print( leftFill, end='' )  # points off for missing end=
    print( oblatt( second, first ) + '-' )  # points off for not using default
    print( leftFill, end='' )  # points off for missing end=
    print( oblatt( first, second ) + '-' )  # points off for not using default
    print( oblatt( second, first, 3 ) )
```

OK to concatenate directly in a print call, not using the variable for the left fill amount
If they figure it out correctly, they can also use "%%s" % ____.
They can also concatenate print( leftFill + oblatt( -- without the end=