CS 1301 CS1 with Robots Summer 2007 – Exam 1

1. Vocabulary Matching: (15 points)
Write the number from the correct definition in the blank next to each term on the left:

_12_Print statement __1_Program __3_Runtime error _14_Semantic error _11_Syntax error __8_Floating-point __6_Integer __4_Integer division __2_Keyword __7_Operator __5_Variable _10_Function _13_function call __9_Type conversion _15_Frame One point for each correct answer.	1. A sequence of instructions that specifies to a computer actions and computations to be performed. 2. A reserved word that is used by the compiler to parse a program; you cannot use things like `if`, `def`, and `while` as variable names. 3. An error that does not occur until the program has started to execute but that prevents the program from continuing. 4. An operation that divides one integer by another and yields an integer. It yields only the whole number of times that the numerator is divisible by the denominator and discards any remainder. 5. A name that refers to a value. 6. A Python data type that holds positive and negative whole numbers. 7. A special symbol that represents a simple computation like addition, multiplication, or string concatenation. 8. A format for representing numbers with fractional parts. 9. An explicit statement that takes a value of one type and computes a corresponding value of another type. 10. A named sequence of statements that performs some useful operation. They may or may not take parameters and may or may not produce a result. 11. An error in a program that makes it impossible to parse (and therefore impossible to interpret). 12. An instruction that causes the Python interpreter to display a value on the screen. 13. A statement that executes a function. It consists of the name of the function followed by a list of arguments enclosed in parentheses. 14. An error in a program that makes it do something other than what the programmer intended. 15. A box in a stack diagram that represents a function call. It contains the local variables and parameters of the function.

2. Vocabulary Matching - Part 2 (15 points)
Write the number from the correct definition in the blank next to each word:

_11_Modulus operator __1_Boolean expression __6_Conditional statement __3_Comparison operator __7_Block _13_Recursion __9_Base case _12_Temporary variable _10_None __8_Guardian __2_Incremental development _14_Multiple assignment _15_Encapsulate __4_Generalize __5_Iteration One point per correct answer.	1. An expression that is either true or false. 2. A program development plan intended to avoid debugging by adding and testing only a small amount of code at a time. 3. One of the operators that compares two values: `==`, `!=`, `>`, `<`, `>=`, and `<=`. 4. To replace something unnecessarily specific (like a constant value) with something appropriately general (like a variable or parameter). 5. Repeated execution of a set of statements using either a recursive function call or a loop. 6. A statement that controls the flow of execution depending on some condition. 7. A group of consecutive statements with the same indentation. 8. A condition that checks for and handles circumstances that might cause an error. 9. A branch of the conditional statement in a recursive function that does not result in a recursive call. 10. A special Python value returned by functions that have no return statement, or a return statement without an argument. 11. An operator, denoted with a percent sign (`%`), that works on integers and yields the remainder when one number is divided by another. 12. A variable used to store an intermediate value in a complex calculation. 13. The process of calling the function that is currently executing. 14. Making more than one assignment to the same variable during the execution of a program. 15. To divide a large complex program into components (like functions) and isolate the components from each other (by using local variables, for example).

3. Write Code (5 points)
Write a function **get_number** that prompts the user to enter a number and returns a floating point value. You do NOT need to check for errors. (Assume the user always enters a valid number).

def get_number():
  num_str = raw_input("Enter a number!")
  num_float = float(num_str)
  return(num_float)
	+1 pts = Function prompts user for number
	+2 pts = Function returns the value
	+2 pts = Function converts string to float.

4. Write Code (5 points)
Write a function **return_largest** that accepts 3 parameters (x,y,z) and returns the largest of the three. For example, **return_largest**(7, -34, 23.8) should return 23.8.

def return_largest(X,Y,Z):
  if (X >= Y) and (X >= Z):
  return(X)
  elif (Y >= X) and (Y >= Z):
  return(Y)
  else:
  return(Z)
	+2pts = function takes 3 parameters
	+2pts = correct value is selected
	+1pts = value is returned.
	
	Test cases to try out on your code:
	return_largest(4,4,5)
	return_largest(5,5,4)

5. Write Code (5 points)
Write a function **draw_a_square()** that will drive your scribbler robot in a square (polygon with 4 equal sides and four 90 degee corners), and beep at each of the four corners. You may assume that turnRight(1, 0.5) will turn your scribbler exactly 90 degrees. You do not need to include the *from myro import* * or *initialize()* calls, and you may assume they have already been done for you. Do NOT use a loop, and do NOT use recursion.

def draw_a_square():
  forward(1,1)
  turnRight(1,0.5)
  beep(1,800)
  <above repeated 3 more times>
	+2 = does a forward,turn, and beep each time, in any order (but order must be consistant)
	+2 = repeats 4 times without for/while loops
	+1 = in a function

```
def n_lines(n):
    if n > 0:
        print "Line!"
        n_lines(n-1)
```

6a. Program Comprehension (1 point)
How many times will the string "Line!" be printed when n_lines is called with n=4?
Number_____Four_____ Note that 0 > 0 is False, so the print statement does not execute when N=0.

6b. Simple Stack Diagram (4 points)
Draw a stack diagram for the function n_lines called with n = 4. (i.e. n_lines(4))
Include the value of any local variables, and remember to start with __main__.

__main__ n_lines: N = 4 n_lines: N = 3 n_lines: N = 2 n_lines: N = 1 n_lines: N = 0	+ 2 points - stack has 5 function calls of n_lines. +1 point - starts with __main__ +1 point - N = 4,3,2,1,0 in the 5 calls.

7. Write Code (4 points)
Write a function with infinite recursion named **run_forever**. Your function should have no parameters, and it should run forever when called (on an ideal computer, in a real computer it would eventaully run out of memory.) You may add a print statement if you wish.

def run_forever() run_forever() # This is a recursive call	+2 points - it's a function +2 points - that calls itself, without a termination clause.

8. Write Code (6 points)
Rewrite the function n_lines from question 6 using a for loop instead of recursion.

def n_lines(N):
  for i in range(N):
  print "Line!"
	+1 pt = it's in a function
	+1 pt = function accepts N as a parameter
	+2 pt = The for statement is correct (executs N times)
	+2 pt = Function has the same behavior as the original: Prints Line! N times.

9. Write Code (8 points)
Rewrite the function n_lines from question 6 and 8 using a while loop instead of recursion or a for loop. You may use **n** as your looping variable.

def n_lines(N)
  while N > 0:
  print "Line!"
  N = N - 1
	+1 pt = it's in a function
	+1 pt = Function accepts N as a parameters
	+2 pts = While statement is correct
	+2 pts = Looping Variable is decremented
	+2 pts = Function has same behavior as the original: Prints Line! N times.

10. Stack Diagram (4 points)
Draw a stack diagram for the code you wrote for problem 9 when n_lines is called with n=4 (e.g. n_lines(4)). Remember to start with __main__ and to show all local variables. If local variables change durring execution, ~~strike them out~~ and show the new value each time they change.

__Main__ n_lines: ~~N = 4~~ ~~N = 3~~ ~~N = 2~~ ~~N = 1~~ N = 0	+ 2 = has 2 frames, __main__ and n_lines +2 = Shows N (or looping variable) decrementing as the while loop runs. (-1 point if they forget the last value of the looping variable that makes the continuation condition turn false.)

11. Python Expression Evaluation (14 points)
Pretend that you are the Python Interpreter (IDLE window). What do you print or return when each of the following statements are entered?

Example: **(7+4) / 2** *Result:* **5**

Example: range(4) *Result:* _ [0, 1, 2, 3]_

1. **(7.0 + 4) / 2** *Result:* *5.5*
2. **7 + 3 / 2** *Result:* *8*
3. **range(4 , 8)** *Result:* *[4,5,6,7]*
4. **range(4 , 8 , 2)** *Result:* *[4,6]*
5. **7.0 > 5.0** *Result:* *True*
6. **7 + 3 / 2 > 8** *Result:* *False*
7. **print "Pumpkin %.3f" %3.1459** *Result:* *Pumpkin 3.146*

Grading: 2 points for each correct answer.

12. Extra Credit (1 point)

What is the name you gave your robot? _____

Grading: Any name gets a point!

13. Extra Credit (2 points)

Where does Python get it's name?

Monty Python's Flying Circus, a British comedy TV show.

14. Extra Credit (3 points)

What are Isaac Asimov's 3 laws of robotics?

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey orders given to it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Grading: 1 point for each law that is close.