ENGIE 1006
Name: _____
UNI: _____
Fall 2016
Exam 1
October 19th, 2016
Time Limit: 70 Minutes

My signature on this line is required. My signature indicates that I understand that I am not permitted to retain any written record, photograph, or facsimile of this test material. I am not permitted to communicate the substance of this test material to other students. My signature on this line is *required* or this test is void and shall not be graded. **I hereby sign here:**

This exam contains 13 pages (including this cover page) and 3 questions.
Total of points is 100.

Show all your work. If you write down an answer without justifying it or explaining your thoughts, you may receive partial or ZERO credit even if the answer is correct.

Put a box around your final answer.

Grade Table (for teacher use only)

Question	Points	Score
Multiple Choice	20	
Short Answers	40	
Writing Codes	40	
Total:	100	

---

1 Multiple Choice (20 pts)

Circle the type of the value that is returned by each of the following statements. If the statement would result in a ValueError, circle 'Error'. Mark your selection unambiguously. To be safe, you may write the letter of your final choice in the space next to the number of the question.

(a) `int('78.3')`

A. int

B. string

C. float

D. error

(b) `float(str(35.888))`

A. int

B. string

C. float

D. error

(c) `int(59.5)`

A. int

B. string

C. float

D. error

(d) `int(str(float(42)))`

A. int

B. string

C. float

D. error

(e) `print(list(range(0, 5)))`

A. [0, 1, 2, 3, 4, 5]

B. [0, 1, 2, 3, 4]

C. [1, 2, 3, 4, 5]

D. [1, 2, 3, 4]

(f)
```
print(15 % 4 == 1)
```

A. True

B. False

(g)
```
a = 3
b = 4
c = 5

a = b
b = c
c = a
print(a, b, c)
```

A. 3 4 5

B. 3 5 3

C. 4 3 5

D. 4 5 4

E. 5 4 4

(h)
```
def my_function(my_input):
    return my_input
    print ("Man I sure do love pancakes")

print(my_function('I could really go for some french toast'))
```

A. I could really go for some french toast

B. Man I sure do love pancakes

C. I could really go for some french toast
Man I sure do love pancakes
None

D. I could really go for some french toast
Man I sure do love pancakes

E. None

(i)

```
def sumToN(n):
total = 0
counter = 0
while (counter <= n):
    counter += 1
    total += counter
return total

print(sumToN(5))
```

A. 10

B. 15

C. 16

D. 21

(j) Which of the following is an invalid variable name?

A. __2016wasPrettyRough

B. Heath__Ledger__Best__Joker2008

C. 100isMyScoreOnThisTest

D. arr__max

2 Short Answers (40 pts)

(a) What is the print value of this code snippet?

```
def foo(a, x):
      res = []
      for i in range(0, len(a)):
           if a[i] == x:
                res.append(i) #append adds the value to the end of the list
           else:
                res.append(x)
      return res

a = [1,2,3,4,5,6,7]
x = 3
print(foo(a, x))
```

Solution: [3, 3, 2, 3, 3, 3, 3]

(b) How many lines will this code snippet print?

```
i = 100
while i >= 0:
  print("my favorite class is 1006")
  i += 1
```

Solution: Infinite or $2^{32} - 101$

(c) What is the print value of this code snippet?

```
b = [7, 8, 9, 10]
a = [1, 2, 3, 4]
for i in range(1, len(a)):
  a[i] += b[i-1]
print(a)
```

Solution: [1, 9, 11, 13]

(d) Ada is attempting to import one of her python files, Division.py, into another one of her files, Calculator.py, so that she might use the functions of Division.py within Calculator.py. The contents of Division.py are:

```
def do_integer_division(x,y):
  return x // y

def do_floating_point_division(x,y):
  return x / y
```

Division.py and Calculator.py are located within the same directory on Adaȧs computer.

Inside of Calculator.py, she wishes to print out the result of performing integer division on 5 with 2 using the functionality of Division.py. To that end, she writes Calculator.py as such:

```
import Division

a=5
B=2
result = do_integer_division(a,b)

print(result)
```

Unfortunately, there is at least one bug in Calculator.py code which causes an error. Rewrite Calculator.py in the space below such that if Calculator.py were run with your modifications, it would work as expected. Do not modify Division.py.

Solution:

```
import Division

a=5
B=2 # typo in this variable name
result = Division.do_integer_division(a,B)

print(result)
```

or

```
from Division import *

a=5
```

```
B=2 # typo in this variable name
    result = do_integer_division(a,B)

print(result)
```

or

```
from Division import do_integer_division

a=5
B=2 # typo in this variable name
result = do_integer_division(a,B)

print(result)
```

(e)
```
def is_greater(a, b):
  if a > b:
  print ("a is bigger than b!")
  elif a == b:
  print ("a is equal to b!")
  else:
  print ("a is smaller than b!")
is_greater(3,6) #[1]
is_greater(2,2) #[2]
is_greater(8,1) #[3]
```

i. Write the printed results for the statements above.

Solution: is_greater(3, 6) – a is smaller than b!
is_greater(2, 2) – a is equal to b!
is_greater(8, 1) – a is bigger than b!

ii. Which statement performs the LEAST comparisons within is_greater()? Circle it below.

A. [1]

B. [2]

C. [3]

(f) Fill in the blank.

```
def selectionSort(a):
    n = len(a)
    for startIndex in range(0, n-1):
        minIndex = startIndex

        for i in range(startIndex+1, n):

            if (a[minIndex] > A[i]):
                minIndex = i

        temp=a[startIndex]
        a[startIndex]=a[minIndex]
        a[minIndex]=temp
```

(g) i. What is the print value of the the following program?

```
input_list = [10, 4, 12, 1, 15, 13, 8, 2, 3]
count = 0
for element in input_list:
    if (element > 10):
        count += 1
print(count)
```

Solution: 3

ii. Write a program that performs equivalent logic on some list 'input_list' as the program in a) but which uses a while loop instead of a for loop. i.e. both the program in a) and in b) will end up printing the same 'count' value for the same list; the count variable should represent the same quantity in both programs.

Solution:

```
input_list = [10, 4, 12, 1, 15, 13, 8, 2, 3]
count = 0
i = 0
while i < len(input_list):
    if input_list[i] > 10:
        count += 1
    i += 1
print(count)
```

(h) Fill in the blanks such that the following code will convert a non-negative integer from user input in a binary number and print that binary number. You can assume that the user will behave properly and not give an input which will result in an error.

```
num = int(input("Convert a non-negative integer to binary: "))
    if num != 0:
        res = ""
    else:
        res = "0"
    while num != 0:
        remainder = num % 2
        num = num // 2
        res = str(remainder) + res
    print(res)
```

(i) Give an integer value n for which foo(n) returns True.

```
def foo(n):
    if n < 100 or n > 999:
        return False
    a = n % 10
    b = n // 10
    return ((a**2==b) and (a + b == 42))
```

Solution: 366

(j) What value of k do you need to guarantee that f(a, k) will return True for any non-empty list 'a' (there are more than one correct answer, you only need to give one). k must be an integer.

```
def f(a, k):
    for i in range(0, len(a)):
    if (a[i] % 2 == 1):
        a[i] -= k
    return  (sum(a) % 2 == 0) #sum(a) sums every element in list a
```

Solution: any odd integer

(k)
```
def mystery(a, b):
    while b:
        c = a
        a = b
        b = c%b
   return a
```

What will the following statements return?

mystery(12, 18)	6
mystery(17, 28)	1

3 Writing Codes (40 pts)

(a) Given two non-negative integers x and y, write a function is_multiple(x, y) which will return True if x is a multiple of y, and False otherwise. x is a multiple of y when there exists a positive integer k such that x = y*k.

Hint: Be careful of handling all edge cases.

```
def is_multiple(x,y):
    if x == 0 or y == 0:
        # the only multiple of 0 is itself
        return x == 0 and y == 0

    return x % y == 0
```

(b) A lonely prime is a positive integer p that is a prime and is not within 15 (inclusive) of the previous lonely prime. 2 is the first lonely prime (since there are no primes before it), but 3, 5, 7, 11, 13, 15 and 17 are not, as they are too close to 2 (so not lonely). 19 will be the next lonely prime.

With this in mind, write a function nth_lonely_prime(n) that takes a non-negative integer n and returns the n-th lonely prime. You may assume that is_prime(n) is already written for you. is_prime(n) takes an integer, n, and returns True if the number is prime, False otherwise.

Sample input and output:

nth_lonely_prime(0) returns 2
nth_lonely_prime(1) returns 19
nth_lonely_prime(2) returns 37

```
def nth_lonely_prime(n):
    curr_lonely_prime = 2
    idx = 0
    while idx < n:
        # has to be 16 or greater than the previous lonely prime
        curr_lonely_prime += 16
        while not is_prime(curr_lonely_prime):
            curr_lonely_prime += 1
        idx += 1

    return curr_lonely_prime
```

Left blank in case you need extra space.

Left blank in case you need extra space.