Exam 1 Solution

1. Define lists as follows: **L_1 = [ ], L_2 = [1, 0], L_3 = [2, 3]**

For each expression below, fill in the value returned when the expression is evaluated.

L_1 and L_2	**L_1**
L_1 or L_2	**L_2**
L_2 and L_3	**L_3**
L_3 and L_2	**L_2**
L_2 or L_3	**L_2**
L_3 or L_2	**L_3**
~L_1	**False**
any(L_1)	**False**
any(L_3)	**True**

Exam 1 Solution

2. Write a function named avg that accepts an arbitrary number of arguments (assumed to be numbers) and returns their average value

```
def avg( *k ):
   return sum(k)/len(k)
```

3. Suppose L is a list of 2-tuples of float objects. Write a **single line** that will sort the list in order of the sum of the entries of each tuple.

Example:

Before sorting: `L = [(1.0,3.2), (-1.0,1.0), (1.0,1.0)]`
After sorting: `L = [(-1.0,1.0), (1.0,1.0), (1.0,3.2)]`

```
L.sort(key = sum)
```

4. Write a list comprehension that produces the same list as the following code:

```
L = []

for k in K:
     if k % 2 == 0:
            L.append(k//2)
```

```
L = [k//2 for k in K if k %2 == 0]
```

or

```
L = [k//2 for k in K[0::2]]
```

5. Write the code for the function get_int(msg) which is intended to input an integer from the user and return that object. Input from the user is obtained by calling input(msg) – msg is the prompt. If an error occurs, print the error message and return; otherwise return the integer. Recall that input() returns a string.

```
def get_int(msg):
   try:
      return int(input(msg))
   except ValueError as e:
      print(e)
```

Exam 1 Solution

6. Suppose L = [R, S, T, U] is a list of sets. Using ordinary set notation, describe the following set:

```
{ x for x in L[0] if all([x in A for A in L[1:])}
```

R ∩ S ∩ T ∩ U

7. Write a **code segment** that will swap the first and second halves of a list. For example, if

L = [1,2,3,4,5], after the code execution L will be [3,4,5,1,2]; if L = [1,2,3,4,5,6], then after the code execution, L will be [4,5,6,1,2,3]

Hint: slices

```
L[:len(L)//2], L[len(L)//2:] = (L[len(L)//2:], L[:len(L)//2])
```

or

```
L = L[len(L)//2:] + L[:len(L)//2]
```

8. Let D be a dictionary whose values are all hashable. Construct a dictionary E that maps distinct values of D to the list of all keys of D that map to the given value.

```
E = { v : [k for k in D if D[k] == v] for v in D.values() }
```

or

```
from collections import defaultdict

E = defaultdict(list)
for k,v in D.items():
   E[v].append(k)
```