1. Show the output from the following program fragments:

```
(a)  for i in range(5):
         print(i*i)

(b)  for d in [3,1,4,1,5]:
         print(d,end=' ')

(c)  for i in range(4):
         print("Hello")

(d)  for i in range(5):
         print(i,2**i)

(e)  for c in ['g','o','o','d','b','y','e']:
         print(c,sep='')

(f)  D = {'Prof':'Tindell','Dept':'CSE','School':'USF','City':'Tampa'}
     for k in D.keys():
         print(k,'is',D[k])

(g)  L = ['To','be','or','not','to','be']
     print(' '.join(L))

(h)  L = ['6','12','2012']
     print('/'.join(L))
     K = L[-1:]+L[:2]
     print('-'.join(K)) "  "

(i)  S = "Elementary my dear Watson"
     print(S.split())

(j)  S = """Well, Stanley,
     isn't this
     a fine
     kettle of fish
     you've gotten me into"""
     print(S.splitlines())

(k)  i = input("Enter an integer: ")
     j = input("Enter another integer: ")
     print("Adding them together we get",i+j)

     Suppose the user enters 25 at the first prompt and 10 at the
     second prompt.  Show the output.

(l)  L = ['to','be','or','not','to','be']
     for k in sorted(L):
         print(k)
```

(m)
```
print(11/4,11//4,11%4)
```

(n)
```
S = 'To be or not to be'
S.replace('be','have been')
print(S)
```

(o)
```
names1 = ['Amir','Barry','Charles','Dao']
print(names[-1][-1])
```

(p)
```
names1 = ['Amir','Barry','Charles','Dao']
loc = names1.index("Edward")
print(loc)
```

(q)
```
names1 = ['Amir','Barry','Charles','Dao']
names2 = [name.lower() for name in names1]
print(names[2][0])
```

(r)
```
confusion = {}
confusion[1] = 1
confusion['1'] = 2
confusion[1] += 1
sum = 0
for k in confusion:
   sum += confusion[k]
print(sum)
```

(s)
```
name = "snow storm"
print(name[6:8])
```

(t)
```
name = "snow storm"
name[5] = 'X'
print(name)
```

(u)
```
for i in range(2):
   print(i,end=' ')
for i in range(4,6):
   print(i,end=' ')
```

(v)
```
x = True
y = False
z = False
if x or y and z:
   print('Yes')
else:
   print('No')
```

```
(w)  x = True
     y = False
     z = False
     if not x or y:
        print(1)
     elif not x or not y and z:
        print(2)
     elif not x or not y or not y and x:
        print(3)
     else:
        print(4)
```

```
(x)  L = [3,1,1,5,3,2,2,4]
     S = set(L)
     print(len(S))
```

2. Suppose we execute the following statemenst

```
D = { }
S = 'one'
L = ['one','two']
T = ('one','two')
```

For each of the following, indicate whether it is a valid Python statement; if it is not valid, explain why.

```
a) D[S] = 'bacon'
b) D[T] = 'bacon'
c) D[L] = 'bacon'
d) D[5] = 'bacon'
```

3. Write a python code segment that

- prompts for and inputs positive integers where the user indicates the end of input by entering the value 0;
- prints out the number of positive integers that were entered, their sum and their average value.

Note: you do not need to store the values in a list.

Example: if the user input is 2 5 1 7 4 2 5 0, then your program should print the following:

```
Number of positive integers: 8
Sum: 26
Average:  3.25
```

4. Write a complete Python program that

- Prompts for and inputs a string named S
- prints the total number of alphabetic characters in S
- for each alphabetic character that appears in S, prints the character, the number of times that character appears in S and its relative frequence (fraction of the total)
- Note: upper and lower case versions of a letter are to be considered the same
- Hint: use tab characters ('\t') to separate the character, count and relative frequency
- Hint: use a dictionary with keys the distinct characters (length-one strings) appearing in S and values the number of times the character appears in S

Your output should match the following example (user input underlined). Except, of course, that the user may enter any string, not just "to be or not to be".

```
Enter a string: to be or not to be

Total number of alphabetic characters:  13

b      2     0.153846153846

e      2     0.153846153846

n      1     0.0769230769231

o      4     0.307692307692

r      1     0.0769230769231

t      3     0.230769230769
```

5. Consider the following function with two parameters, both strings:

```
def  mystery(s,p):
   i = s.find(p)
   if i == -1:
      return False
   while True:
      old_i = i
      i = s.find(p,i+1)
      if i == -1:
         return False
      elif i < old_i+len(p):
         return True
```

Describe in words what this function does.   That is, complete the following sentence:

mystery(s,p) returns True if and only if ____

Information for problems 6 and 7

The time of day using the 12-hour format consists of the hour, a colon, the minute, a space and one of am or pm, where the hour is an number between 1 and 12 and the minute is a number between 0 and 59.

The time of day using the 24-hour format consists of the hour, a colon and a minute. The hour is a number between 0 and 23 and the minute is a number between 0 and 59.

Do not do input validation. That is, you may assume the user always provides valid strings.

6. Write a complete Python program that prompts for and inputs a time in 24-hour format, then prints the same time in 12-hour format

Example: if the user enters 20:32, the program prints 8:32 pm.

7. Write a complete Python program that prompts for and inputs a time in 12-hour format and prints the same time in 24-hour format

Example: if the user enters 4:15 pm, the program prints 16:15.

8. Write a Python program that inputs a string containing only lower-case letters and spaces, then constructs a dictionary whose keys are the distinct words in the string and whose value for a word is the number of times the word appears in the string. After that, it prints the contents of the dictionary by printing for each word in the string a line containing the word, a tab and the number of times the word appears in the string. You must print the lines in ascending alphabetic order of the words.

For example, suppose the string is

```
bill got a bill for a bell
```

Then the output should look like this:

```
a     2
bell  1
bill  2
for   1
got   1
```