CS1026 Review/Practice Questions – Quiz 3

1. Consider the following Python program and then answer the questions following.

```
1.          MAX_STARS = 40
2.
3.          # Read the data values from the user.
4.          values = []
5.          inputStr = input("Enter a value (blank to quit): ")
6.          while inputStr != "" :
7.             values.append(float(inputStr))
8.             inputStr = input("Enter a value (blank to quit): ")
9.
10.         # Identify the largest value.
11.         largest = max(values)
12.         # Display the bars.
13.         for i in range(0, len(values)) :
14.             print("*" * round(values[i] / largest * MAX_STARS))
```

1.a. What is a named constant in the program? **MAX_STARS**
1.b. What type of value is the user expected to enter in line 5? **Float**
1.c. What kind of data structure is being used in the program? **List**
1.d. What variable references that data structure? **values**

2. The following function computes the number of digits in an integer. For example, if the parameter `n` has a value of 113, then the function should return 3. Similarly, if `n` has a value of 4, then it should return 1. The function has 4 logic mistakes; find them and correct them.

```
def digits(n) :
   if abs(n) < 10 :
      return 0                          # return 1
   else :
       num = 0
       while abs(n) > 10:               # >9 or >= 10
           num = num + 1
           n = n % 10                   # n == n // 10
       return num                       # return num+1
```

3. The following main program makes use of the function `digits` defined above. Assume that the function digits has been modified to work correctly. The main program loops repeatedly asking the user to enter an integer or the letter "Q". Anything entered other than a "Q" or an integer is to cause an exception to be raised – a `ValueError` exception. Fill the in blanks with correct code to complete the main program.

```
def main() :
    sg = ""
    while sg != "Q" :
        try :
            sg = input("Enter an integer (to terminate, enter Q: ")
            if sg != "Q" :
                num = int(sg)
                ndig = digits(num)
                print("The number of digits in "+str(num)+" is "+str(ndig))
        except  ValueError :
            print("Input error - retry.")
```

4. Consider the following Python classes and then answer the questions following.

```
#-------------------
## Represent an employee with a name and salary.
class Employee :

   def __init__(self, name, salary) :
      self._name = name
      self._salary = salary

   def __repr__(self) :
      return self._name + " has a salary of %.2f" % self._salary

#-------------------
## Represent a manager with a department.
class Manager(Employee) :

   def __init__(self, name, salary, department) :
      super().__init__(name, salary)
      self._department = department

   def __repr__(self) :
      return self._name + " has a salary of %.2f" % self._salary + \
               " and manages the " + self._department + " department"

#-------------------
## Represent an executive.
class Executive(Manager) :

   def __repr__(self) :
      return self._name + " has a salary of %.2f" % self._salary + \
               " and is the executive for the " + self._department + " department"
```

4.a. What is the constructor for the class `Manager`?

```
def __init__(self, name, salary, department) :
```

4.b. What class is the class `Manager` a subclass of? **`Employee`**

4.c. What class is the class `Executive` a subclass of? **`Manager`**

4.d. If you create an object of the class `Manager`, what instance variables would it have?

`_name, _salary, _department`

4.e. If you create an object of the class `Executive`, what instance variables would it have?

`_name, _salary, _department`

4.f. What does `super()` do in the class `Manager`?

Invokes the constructor in the class Employee

4.g. Write a method for the class `Manager`, that will get the department name of a manager.

```
def getDept(self) :
   return self._department
```

4.h. Consider the following lines of code.

```
emp = Employee("John Smith", 45000.00)

print(man.getDepartment())
```

What happens when you run the code?

The first line will create an object of the Employee class – only. The second line will cause an error.

Write a main method that will make use of these classes. It should a) create a Manager object for "Mickey Mouse", who manages the "Entertainment" department and has a salary of $83,000, b) create and object for "Walt Disney" who is the executive for the "Entertainment" department and has a salary of $195,000, c) print out each object.

```
aMan = Manager("Mickey Mouse", 83000.00, "Entertainment")
theExec = Executive("Walt Disney", 195000.00, "Entertainment")
print(aMan)
print(theExec)
```

Define a class ExecutiveAssistant for employees that work as the assistant to an executive in the company. The class definition should include the name of the executive that they work for.

```
class ExecutiveAssistant :

    def __init__(self, name, salary, department, exec) :
    self._name = name
    self._salary = salary
    self._department = department
    self._execBoss = exec
```