You can find multiple choice review questions (when they are available) in D2L under:
`Assessments->Quizzes`

For all questions, unless otherwise specified assume that there are no syntax errors in any programs or program fragments.

Short answer (code writing):

Begin with the starting program that randomly initializes a 1D list (which is a queue containing items waiting to be service) of 5 – 15 elements. Most elements will be 'empty' (a space). The next most frequent element is a regular item (a period) in the queue. The final type of element is a priority element (exclamation mark).

The starting function will scan the queue and if it's not empty it will service waiting elements one at a time on a first-come, first served basis (F.I.F.O.). Elements on the left side of the list (lower index are ones that are 'earlier'). The exception to the FIFO servicing are the priority elements which are always service before regular items. Elements of the same 'type' (i.e. priority or regular) are served on a FIFO basis. Each scan of the list will allow up to one element to be serviced and then the list is displayed. The list will continued to be scanned until it is all empty whereby the end program message will appear ("Queue has now been emptied")

```
MIN_PERCENT = 1
MAX_PERCENT = 100
MIN_ELEMENTS = 5
MAX = 15
EMPTY = " "
REGULAR = "."
PRIORITY = "!"
ERROR = "?"

# Creates a list of 1 - 100 elements each element a string of length 1
def createList():
    #Declare variables for function
    aList = []
    size = -1
    i = -1

    size = random.randint(MIN_ELEMENTS,MAX)
    i = 0
    while (i < size):
        aList.append(generateElement())
        i = i + 1
    return(aList)
```

```
def display(aQueue):
    #Declare variables for function
    i = -1
    size = len(aQueue)

    #Numbering the columns
    i = 0
    while (i < size):
        print(" %d" %(i), end ="")
        i = i + 1
    print()

    #Display row of lines above list
    i = 0
    while (i < size):
        print(" -", end ="")
        i = i + 1
    print()
    #Display list with bounding line appearing before each element
    i = 0
    while (i < size):
        print("|%s" %(aQueue[i]), end ="")
        i = i + 1
    print("|")

    #Display row of lines below list
    i = 0
    while (i < size):
        print(" -", end ="")
        i = i + 1
    print("\n")

def generateElement():
    #randdint: both min and max value are included in the possible
    #randomly generated values
    dieRoll = random.randint(MIN_PERCENT,(MAX_PERCENT))
    if ((dieRoll >= MIN_PERCENT) and (dieRoll <= 50)):
        aChar = EMPTY
    elif ((dieRoll > 50) and (dieRoll <= 85)):
        aChar = REGULAR
    elif ((dieRoll > 85) and (dieRoll <= 100)):
        aChar = PRIORITY
    else:
        aChar = ERROR
    return (aChar)
```

```
def start():
    aQueue = createList()
    display(aQueue)
    notEmpty = isOccupied(aQueue)
    while (notEmpty == True):
        input("Hit enter to continue")
        remove(aQueue)
        notEmpty = isOccupied(aQueue)
        display(aQueue)

    print()
    print("Queue has now been emptied")
    print("-")

start()
```

Write the code for the following functions:

`def isOccupied(aQueue):`

- **Parameters:** `(list)`: the list which will be scanned in order to determine if it's empty (all spaces) or it has at least one occupied element (non-space character).
- **Return value:** `(Boolean)`: Answers the question whether or not the list contains any elements needed to be serviced.
- **Functionality:** Scans the list parameter. Scanning will stop either when the end of the list has been reached without encountering a non-space character (result of the scan is False because it's false that the list is occupied) or one of the waiting elements has been scanned (result of the scan is True). When scanning ends the state of the scan will be returned to the caller.

`def remove(aQueue):`

- **Parameters:** `(list)`: it's the list which from which a waiting element will be removed.
- **Return value:** `(nothing)`:
- **Functionality:** Exactly one element is removed from the list each time this function has been called. It's never zero elements because this function is only called if it's not empty. The previously describe method (priority-based FIFO queue) is used to determine which element is removed.